

# **Furnace User Manual**

*for version 0.6.8.3*

# authors

---

- brickblock369
- cam900
- DeMOSic
- Electric Keet
- freq-mod
- host12prog
- Lunathir
- tildearrow

special thanks to ZoomTen for providing tools which assisted in the production of this document!

copyright © 2024 tildearrow and other authors.

this documentation is under the [Creative Commons Attribution 3.0 Unported](https://creativecommons.org/licenses/by/3.0/) (<https://creativecommons.org/licenses/by/3.0/>) license.

you may reproduce, modify and/or distribute this documentation provided this copyright notice (including license and attribution) is present and any necessary disclaimers whether modifications have been made.

this documentation is provided as-is and without warranty of any kind.

this manual is written for version 0.6.8.3 of Furnace.  
it may not necessarily apply to previous or future versions.

# contents

---

1. introduction .....	7
1. concepts and terms .....	9
2. frequently asked questions .....	11
3. glossary of common terms .....	16
4. hexadecimal .....	21
5. quick start guide .....	25
2. interface .....	44
1. asset list .....	46
2. UI components .....	51
3. effect list window .....	57
4. export .....	60
5. file formats .....	63
6. keyboard .....	65
7. menu bar .....	72
8. orders .....	76
9. play/edit controls .....	78
10. settings .....	82
11. song info .....	97
3. pattern .....	99
1. effect list .....	104
4. instrument editor .....	112
1. ADPCM-A instrument editor .....	122
2. ADPCM-B instrument editor .....	123
3. AY-3-8910 instrument editor .....	124
4. AY8930 instrument editor .....	125
5. beeper instrument editor .....	126
6. Bifurcator instrument editor .....	127
7. Namco C140 instrument editor .....	128
8. Namco C219 instrument editor .....	129
9. C64 SID instrument editor .....	130
10. Dave instrument editor .....	132
11. Ensoniq ES5506 instrument editor .....	133
12. FDS instrument editor .....	134
13. ESFM instrument editor .....	135
14. OPL FM synthesis instrument editor .....	139
15. OPLL FM synthesis instrument editor .....	142
16. FM (OPM) instrument editor .....	145
17. FM (OPN) instrument editor .....	148
18. FM (OPZ) instrument editor .....	151
19. Irem GA20 instrument editor .....	154
20. Game Boy instrument editor .....	155
21. GBA DMA instrument editor .....	157
22. GBA MinMod instrument editor .....	158
23. Konami K007232 instrument editor .....	159
24. K053260 instrument editor .....	160
25. Atari Lynx instrument editor .....	161
26. MSM5232 instrument editor .....	163

27. MSM6258 instrument editor .....	164
28. MSM6295 instrument editor .....	165
29. MultiPCM instrument editor .....	166
30. Namco 163 instrument editor .....	168
31. Nintendo DS instrument editor .....	169
32. NES instrument editor .....	170
33. NEC PC Engine instrument editor .....	172
34. Commodore PET instrument editor .....	173
35. Pokémon Mini/QuadTone instrument editor .....	174
36. Atari POKEY instrument editor .....	175
37. PowerNoise instrument editor .....	176
38. Sega PSG instrument editor .....	178
39. PV-1000 instrument editor .....	179
40. Capcom QSound instrument editor .....	180
41. Ricoh RF5C68 instrument editor .....	181
42. Philips SAA1099 instrument editor .....	182
43. Amiga/PCM sound source instrument editor .....	183
44. Konami SCC/Bubble System WSG instrument editor .....	184
45. SegaPCM instrument editor .....	185
46. SID2 instrument editor .....	186
47. SID3 instrument editor .....	188
48. SM8521 instrument editor .....	192
49. SNES instrument editor .....	193
50. tildearrow Sound Unit instrument editor .....	195
51. T6W28 instrument editor .....	197
52. TED instrument editor .....	198
53. Atari TIA instrument editor .....	199
54. VERA instrument editor .....	200
55. Commodore VIC instrument editor .....	201
56. Virtual Boy instrument editor .....	202
57. VRC6 instrument editor .....	203
58. Watara Supervision instrument editor .....	204
59. wavetable synthesizer .....	205
60. WonderSwan instrument editor .....	207
61. Namco WSG instrument editor .....	208
62. X1-010 instrument editor .....	209
63. YMZ280B instrument editor .....	210
5. wavetables .....	211
6. samples .....	216
7. systems .....	221
1. 5E01 .....	224
2. Commodore Amiga .....	226
3. General Instrument AY-3-8910 .....	228
4. Microchip AY8930 .....	230
5. Bifurcator .....	232
6. Bubble System WSG .....	233
7. Namco C140 .....	234
8. Namco C219 .....	235
9. Commodore 64 .....	236
10. Generic PCM DAC .....	239
11. Dave .....	240
12. Ensoniq ES5506 (OTTO) .....	242



13. ESS ESFM .....	244
14. Famicom Disk System .....	247
15. Irem GA20 .....	248
16. Game Boy .....	249
17. Game Boy Advance .....	251
18. Konami K007232 .....	252
19. Konami K053260 .....	253
20. Atari Lynx/MIKEY .....	254
21. Nintendo MMC5 .....	255
22. OKI MSM5232 .....	256
23. OKI MSM6258 .....	257
24. OKI MSM6295 .....	258
25. Namco 163 (also called N163, Namco C163, Namco 106 [sic], Namco 160 or Namco 129) .....	260
26. Namco WSG / Namco C15 / Namco C30 .....	262
27. Nintendo DS .....	263
28. NES .....	264
29. Yamaha OPL .....	268
30. Yamaha YM2413/OPLL .....	271
31. Yamaha OPZ (YM2414) .....	273
32. PC Engine/TurboGrafx-16 .....	276
33. PC Speaker .....	278
34. Commodore PET .....	280
35. Pokémon Mini .....	281
36. POKEY .....	282
37. PowerNoise .....	284
38. Casio PV-1000 .....	285
39. Capcom QSound (DL-1425) .....	286
40. Ricoh RF5C68 .....	287
41. Philips SAA1099 .....	288
42. Konami SCC/SCC+ .....	289
43. SegaPCM .....	290
44. SID2 .....	291
45. SID3 .....	293
46. Sharp SM8521 .....	297
47. TI SN76489 (e.g. Sega Master System) .....	298
48. Super Nintendo Entertainment System (SNES) / Super Famicom .....	300
49. tildearrow Sound Unit .....	306
50. Toshiba T6W28 .....	308
51. MOS Technology TED .....	309
52. Atari 2600 (TIA) .....	310
53. VERA .....	318
54. Commodore VIC-20 .....	319
55. Virtual Boy .....	320
56. Konami VRC6 .....	322
57. Watara Supervision .....	323
58. WonderSwan .....	324
59. Seta/Allumer X1-010 .....	326
60. Yamaha YM2151 .....	328
61. Yamaha YM2203 (OPN) .....	330
62. Yamaha YM2608 (OPNA) .....	334
63. Neo Geo/Yamaha YM2610 .....	338

64. Taito Arcade/Yamaha YM2610B .....	342
65. Yamaha YM2612 .....	346
66. Yamaha YMU759 .....	350
67. Yamaha YMZ280B (PCMD8) .....	351
68. ZX Spectrum beeper .....	352
8. advanced .....	353
1. channels .....	354
2. oscilloscope (per-channel) .....	356
3. chip manager .....	360
4. clock .....	362
5. command line usage .....	363
6. comments .....	366
7. compatibility flags .....	367
8. find/replace .....	368
9. grooves .....	372
10. input latch .....	375
11. log viewer .....	376
12. memory composition .....	377
13. mixer .....	378
14. operation mask .....	380
15. oscilloscope .....	381
16. pattern manager .....	382
17. piano / input pad .....	384
18. register view .....	386
19. statistics .....	387
20. user systems .....	388
21. oscilloscope (X-Y) .....	390
9. guides .....	391
1. choosing emulation cores .....	392
2. AY-3-8910 / AY8930 / SAA1099 envelope guide .....	394
3. using samples with limited playback rates .....	396
4. using OPLL patch macro .....	397
5. tuning samples .....	398

# introduction

---

Furnace is a tool which allows you to create music using sound chips ("chiptune"), most from the 8/16-bit era.

it has a large selection of features and sound chips. from the NES, SNES and Genesis to ES5506, VIC-20 or even Arcade, Furnace has most likely covered your target with many presets to choose from.

every chip is emulated using many emulation cores, therefore the sound that Furnace produces is authentic to that of real hardware.

## quick start

---

if you just want to jump right in and get going, we recommended you go through the [quick start guide](#)<sup>25</sup> first. it should get you familiar enough with the program to comprehend the rest of the documentation.

## hexadecimal

---

Furnace uses hexadecimal (abbreviated as "hex") numbers frequently. see [this guide](#)<sup>21</sup> for a crash course.

## interface

---

Furnace uses a music tracker interface. think of a table with music notes written on it. then that table scrolls up and plays the notes.

for an introduction to a tracker interface, see [tracker concepts and terms](#)<sup>9</sup> before using Furnace. there's also a [glossary of common terms](#)<sup>16</sup>.

Furnace uses a flexible windowing system which you may move around and organize. see [2-interface](#)<sup>44</sup> and [3-pattern](#)<sup>99</sup> for more information.

once familiar with the tracker, look to [9-guides](#)<sup>391</sup> for useful techniques.

## tutorial?

---

[How to Learn Chiptune Trackers](https://www.youtube.com/watch?v=Q37XuOLz0jw) (https://www.youtube.com/watch?v=Q37XuOLz0jw) : video tutorial created by Button Masher. covers the basic mechanics of chiptune tracking using Furnace for demonstration.

[Furnace Tutorials](https://youtube.com/playlist?list=PLCELB6AsTZUnwv0PC5AAGHjvg47F44YQ1) (https://youtube.com/playlist?list=PLCELB6AsTZUnwv0PC5AAGHjvg47F44YQ1) : video tutorials created by Spinning Square Waves. be noted that these may not apply to the current version.

## for more information

---

see the [frequently asked questions](#)<sup>11</sup>.

# concepts and terms

---

- a **song** (also called **module**) is a file for a tracker that contains at least one **subsong**.
- each Furnace song involves at least one **chip**, an emulation of a specific audio processor.

## tracking

---

the **pattern view**<sup>99</sup> is similar to spreadsheet that displays the following:

- each labeled column represents a **channel** of sound provided by the chips in use.
- each **note** starts a sound playing. within a channel, only one note can play at a time.
- each note is assigned an **instrument**<sup>112</sup> which describes what it will sound like.
- an **effect** is a command that changes some aspect of playback. it can alter note pitch, volume, timing, and more.
- an instrument **macro** is an automated sequence of effects that applies to every note of that instrument.
- during playback, the **playhead** moves down, scrolling through the pattern view, triggering the notes that it encounters.

## structure

---

the **order list** is a smaller spreadsheet showing the overall song structure.

- a song is made up of a list of orders.
- an **order** is a set of numbered patterns used for each channel.
- each channel has its own unique list of patterns.
- each **pattern** contains note and effect data for that channel only.
- patterns may be used multiple times in the order list. changing a pattern's data in one order will affect the same pattern used in other orders.
- each pattern is made of the same number of rows as seen in the tracker view.
- during playback, the **playhead** moves down as described previously. when it reaches the end of the pattern view, it will go to the next order.
- if the last order is reached and the playhead reaches the end of the pattern view, it will go back to the beginning of the song.

## time

---

- during playback, each **row** lasts a number of ticks determined by the song's **speed** value(s).
- a **tick** is the smallest measure of time to which all note, effect, and macro times are quantized.

# sound

---

sound chips have different capabilities. even within the same chip, each channel may have its own ways of making sound.

- some channels use one or more waveform **generators** (sine, square, noise...) to build up a sound.
- of special note are **FM (frequency modulation)** channels, which use a number of generators called **operators** that can interact to make very complex sounds.
- some channels use **samples**<sup>216</sup> which are (usually) recordings of sounds, often with defined loop points to allow a note to sustain.
- some channels use **wavetables**<sup>211</sup>, which are very short samples of fixed length that automatically loop.

# frequently asked questions

## general

### how do I use Furnace?

the [quick start guide](#)<sup>25</sup> and [concepts list](#)<sup>9</sup> are great first steps in getting used to the tracker way. after that, try exploring the demo songs to learn more about how to get the sounds you want.

### is there an Android build?

yes and no. Furnace can be successfully built for Android and it even has some rudimentary touch UI support, but it's extremely unfinished and extremely unsupported. there are many good reasons that it's not in the official releases yet.

### is there an iOS build?

nope:

- in order to test any iOS app on an actual device, the developer must pay a hefty 99 USD per year to just enroll for Apple Developer Program. Furnace is free and non-profit.
- developing for iOS requires an Apple computer and an Apple device for testing.
- even if an alternate store were used, you would still need a method to sideload an app to your device. jailbreaking an iOS device isn't an easy task.

for these reasons, an iOS build will never happen until a major shift occurs in Apple's policies.

### what are the system requirements for Furnace to run?

Furnace is developed to be able to run on operating systems as old as Windows XP, on downright ancient (Pentium III) 32-bit CPUs, with Linux ABIs dating as old as 2015. safe to assume any computer made past 2010, no matter how underpowered at time of release, will be able to run Furnace perfectly.

	MINIMUM	RECOMMENDED
OS	Windows XP SP3 32-bit, Ubuntu 16.04, Android 6.0, or MacOS 10.9	Windows Vista SP2 64-bit, Ubuntu 20.04, Android 11, or MacOS 10.15
CPU	Intel Pentium 3, AMD Athlon 64, or 32-bit ARMv7	fourth-generation Intel Ivy Bridge, AMD Zen, or AArch64 ARM CPU*
GPU	any (software rendering)	any GPU supporting OpenGL 3.0, OpenGL ES 2.0 or DirectX 11
storage	25MB (program) + 200MB (config and backups)	60MB (program and included assets) + 1GB (config and backups)
RAM	768 MB	2 GB
display	800x600	1280x720

some emulation cores may have higher processor requirements:

- ESFMu: 2.0GHz Skylake/Zen, or 2.9GHz Cortex-A76/Snapdragon 845/Apple M1
- Nuked-OPM and reSID: 3.0GHz Skylake/Zen, or 2.9GHz Cortex-A76/Snapdragon 845/Apple M1
- reSIDfp: 4.0GHz Skylake/Zen 2, or Cortex-A710/Snapdragon 870/Apple M1

## **will Furnace ever have a piano roll or DAW interface?**

there are no plans for this. Furnace is a tracker, and changing the interface would involve a long and painful rewrite.

## **I've lost my song – what do I do?**

Furnace keeps backups of the songs you've worked on before. go to **file > restore backup**.

## **workflow**

---

### **how do I compose in 6/8, 5/4, or other time signatures?**

set the pattern length to a multiple of the number of beats (6 or 5 as mentioned above). don't forget to change the row highlight values to match!

### **how do I do triplets, quintuplets, or other tuplets?**

there are two common methods:

- use delay commands (EDxx) to offset notes into their correct places. this is good for the occasional set of tuplets, but if you expect to use a lot of them...
- plan ahead for the song to have them by making your pattern length a multiple of that number. remember to adjust row highlight values to match.

depending on the tempo of the song, it may only be possible to get perfectly even tuplets by changing the tick rate. mind that this may hinder playback in games or sound engines that use the vertical blank interval for their timing.

### **why do certain notes not play low enough, high enough, or in tune?**

each chip has its own set of limitations regarding what frequencies it can play. if these limits are likely to be found in normal tracking, they'll be mentioned in [that chip's documentation](#).

### **can I add effects to the output like EQ or reverb?**

not yet, but it's in the early development stage.

## **chips**

---

### **will Furnace support the Sony PlayStation?**

it's in the plans, with no target date.



Sony Playstation uses a stereo 24 channel sample-based synthesizer using a custom ADPCM-like format for samples and providing a 512 kilobytes of RAM for these.

## **will Furnace support the Sega Saturn?**

it also is in the plans, with no target date.

the Sega Saturn uses an extremely complicated Yamaha YMF292 sound chip, employing both FM and sample based-synthesis with 32 channels (or rather, FM operators). each channel can frequency modulate another channel, which lowers the polyphony count according to how many operators are used per voice, with no clear concept of operator routing.

## **will Furnace support the Nintendo 64?**

the N64 lacks any form of audio synthesizer chip. many games use MIDI or XM or other such formats internally, but everything is mixed in software and sent to a simple stereo DAC.

## **will Furnace support this obscure PCM-only chip?**

probably not, as with very few exceptions these are effectively all the same.

## **will Furnace support the Roland MT-32?**

no. MT-32 is used with MIDI in 99.999% of situations. it lacks a direct register interface.

also, Furnace is not a MIDI tracker....

# **importing**

---

## **will Furnace import MIDI files?**

nope. Furnace is not a MIDI tracker.

## **but why?**

trackers aren't exactly optimal for MIDI, compared to dedicated MIDI editors. for example, trackers are designed to only have a single note on/off, not an array of 128 different notes to turn on/off in the same tick and channel. even if one were to make a new 16-channel system which maps to each MIDI channel, true polyphony in the same channel would still be impossible. moreover, dynamic channel allocation and handling effect states between channel would get messy very fast.

## **why does this imported file sound wrong?**

there are fundamental differences between formats that cannot be directly translated. an import should always be considered the starting point of a conversion, not a final product.

## **can I import VGM or NSF?**

nope. it's a feature that's been requested many times, but there are no plans to implement it. VGM files are raw register dumps, and NSF/KSS/SID files are RAM programs with driver code and song data optimized for said driver. neither of them translate nicely into a pattern view.

for NSF import, you can use [a modified version of FamiTracker called NSFImport](http://rainwarrior.ca/projects/nes/nsfimport.html) (<http://rainwarrior.ca/projects/nes/nsfimport.html>) and then import the resulting .ftm into Furnace. it's all speed 1 though, so don't expect any songs to be nicely laid out with instruments and all.

## **how can I use an SF2 soundfont?**

one way is to use [OpenMPT](https://openmpt.org/) (<https://openmpt.org/>) to open the SF2 file, and save WAV files from there. [Polyphone](https://www.polyphone.io/) (<https://www.polyphone.io/>) is another way.

## **how do I import instruments from this SNES game?**

use [split700](https://github.com/gocha/split700) (<https://github.com/gocha/split700>) to extract the BRR samples from an SPC. there is presently no way to import envelopes or other parameters.

## **how do I import instruments from this Sega Genesis game?**

extract FM patches from a VGM file using [vgm2pre](https://github.com/vgmtool/vgm2pre) (<https://github.com/vgmtool/vgm2pre>) or similar tools like [OPN2 Bank Editor](https://github.com/Wohlstand/OPN2BankEditor) (<https://github.com/Wohlstand/OPN2BankEditor>) or [YM2608 Tone Editor](https://github.com/rerrahkr/YM2608-Tone-Editor) (<https://github.com/rerrahkr/YM2608-Tone-Editor>) . bear in mind that these are only the parameters for the FM synth, and the way the instrument is heard in-game may include pitch bends or other effects that can't be extracted.

for PSG instruments, see the next question.

## **how do I import instruments from this NES/SMS/GB/C64/etc. game?**

PSG chips (such as those in the systems mentioned) don't have any inherent concept of instruments or patches. all of that is handled in software, and each sound driver has its own way of doing things. generally, the only option is to recreate the instrument from scratch.

# **exporting**

---

## **will Furnace export MIDI files?**

nope. Furnace is not a MIDI tracker.

## **why does this exported VGM sound weird when I play it in other software?**

just as Furnace offers a choice of emulation cores, VGM players may use different cores with varying degrees of accuracy. also, some aspects of a song may not be supported by the VGM format, such as chip clock speeds.

## **why does this exported DMF sound wrong in DefleMask?**

while Furnace did start life as a DMF player, it's grown in functionality quite a bit, and many Furnace features simply don't exist in that format. there are also cases where the emulation cores in DefleMask sound different from those available in Furnace.

## **when will Furnace be able to export to a ROM for a particular system or an emulated music format?**

each system will need its own method of converting Furnace songs into code that can be played back on hardware. this requires writing a driver for the hardware in question, which is no small task. that having been said, there are several efforts in progress, both for direct export from Furnace itself and external converters such as [furSPC](https://github.com/AnnoyedArt1256/furSPC) (<https://github.com/AnnoyedArt1256/furSPC>) , [furNES](https://github.com/AnnoyedArt1256/furNES) (<https://github.com/AnnoyedArt1256/furNES>) , and [furC64](https://github.com/AnnoyedArt1256/furC64) (<https://github.com/AnnoyedArt1256/furC64>) .

Furnace 0.7 is expected to support exporting ROM for SNES, C64 and Game Boy. there is no ETA for this version.

## **can Furnace export MP3/OGG/FLAC files?**

not presently. for now, use an external converter such as FFmpeg.

## **other**

---

if a question isn't answered within this manual, check in the [GitHub Discussions](https://github.com/tildearrow/furnace/discussions) (<https://github.com/tildearrow/furnace/discussions>) to see if it's answered there, and post if needed.

# glossary of common terms

---

**2-op, 3-op, 4-op...**: the number of FM operators used to generate a sound. more operators allow for more complex sounds.

**ADPCM**: adaptive differential pulse code modulation. this is a variety of DPCM with a more complex method of storing the amplitude differences.

**ADSR**: attack, decay, sustain and release. these are elements that comprise a basic envelope.

**algorithm**: the way in which the operators in an FM instrument interact.

- when two operators connect to the same point, their sounds are added together.
- when two operators are connected left to right, the left is the modulator and the right is the carrier sound that is modified.

**asset**: an instrument, wavetable or sample.

**bit**: a single binary on-off value.

**bitbang**: to achieve PCM sound by sending a rapid stream of volume commands to a non-PCM channel.

**bit depth**: the number of bits used for each sample in a PCM stream. 16-bit is considered high quality; lower bit depths introduce more noise and distortion, especially for quieter samples.

- the "1-bit DPCM" sample type has 7-bit values, but each sample is represented by only 1 bit in indicating whether its value differs from the previous sample by one step up or down.

**bitmask**: a set of bits which represent individual single-bit toggles or groups representing small numbers. these are explained fully in the [hexadecimal primer](#)<sup>21</sup>.

**BRR**: a lossy sample format used by the SNES. it has a fixed compression ratio; groups of 32 bytes (16 samples) are encoded in 9 bytes each.

- usually stored in .brr files.

**clipping**: when a sample or playback stream exceeds the maximum or minimum values. this can cause audible distortion.

- this often occurs when a sample is amplified too much.
- it can also occur during playback if too much sound is being added together at once. in some cases the mixer can be used to reduce the volume. if this doesn't work, the clipping is caused within the chip's own mixing, and the only solution is to reduce the volumes of the notes being played.

**clock rate**: the timing at which a chip operates, expressed as cycles per second (Hz).

- changing this may change aspects of how some chips work, most notably pitch.
- some chips cannot operate at anything other than their designed clock rate.

**cursor** (1): the marker of input focus. anything typed will happen at the cursor's location.

**cursor** (2): the pointer controlled by a mouse or similar input. clicking when the cursor(2) is in a valid area will place the cursor(1) there.

**DAC:** digital analog converter. this converts a digital representation of sound into actual output.

**.dmf:** DefleMask Module File.

- *Furnace:* .dmf files may be read, and compatibility flags will be set to make them play as accurately as possible, but there may still be glitches.
- *Furnace:* .dmf files may be saved, but full compatibility isn't guaranteed and many features will be missing. this isn't recommended unless absolutely necessary.

**.dmp:** DefleMask Preset. an instrument file.

**.dmw:** DefleMask Wavetable. a wavetable file.

**DPCM:** differential/delta pulse code modulation. this is a variety of PCM that stores each amplitude as its difference from the previous.

**duty cycle:** usually called *pulse width*. in a pulse wave, this is the ratio of the high part to the high and low combined.

**feedback:** in FM instruments, this adds some of an operator's output into itself to create complex harmonics.

- in the algorithm view, an operator with a circle around it is capable of feedback.

**FM:** frequency modulation. this is a method of generating sound that uses one operator's amplitude to modify another operator's frequency.

- the FM in Yamaha chips is more accurately called *phase modulation*, which uses a different method of computation to achieve similar results.

**.ftm:** FamiTracker Module.

**.fui:** a Furnace instrument file.

**.fur:** a Furnace module file.

**.fuw:** a Furnace wavetable file.

**hard-pan:** sounds can only be panned to the center, 100% left, or 100% right. this often appears in the instrument editor's panning macro as on/off toggles for the left and right channels.

**Hz:** hertz. a unit representing divisions of one second. 1 Hz means once per second; 100 Hz means one hundred times per second. also, *kHz* (kilohertz, one thousand per second) and *MHz* (mega hertz, one million per second).

**interpolate:** to fill in the area between two values with a smooth ramp of values in between.

- some sample-based chips can interpolate, filtering out unwanted harmonics.

**.it:** Impulse Tracker module.

**ladder effect:** an inaccurate yet common term for the DAC distortion that affects some Yamaha FM chips.

**LFO:** low frequency oscillator. a wave with a slow period (often below hearing range) used to alter other sounds.

**LFSR:** linear-feedback shift register. a method to generate pseudo-random noise that loops, also known as "periodic noise". within a sequence of on-off bits, it does math to combine the bits at specified locations called "taps", then shifts the whole sequence and adds the resulting bit on the end, guaranteeing a different state for the next pass. depending on the locations of the taps, different lengths of noise loops are generated; for short loops, this will affect their tone.

**macro:** a sequence of values automatically applied while a note plays.

**noise bass:** the technique of using a PSG's periodic noise generator with a very short period to create low-frequency sounds.

**normalize:** to adjust the volume of a sample so it is as loud as possible without adding distortion from clipping.

**operator:** in FM, a single oscillator that interacts with other oscillators to generate sound.

**oscillator:** a sine wave or other basic waveform used as sound or to alter sound.

**PCM:** pulse code modulation. a stream of data that represents sound as a rapid sequence of amplitudes.

**period:** the length of a repeating waveform. as frequency rises, the period shortens.

**periodic noise:** an approximation of random noise generated algorithmically with an LFSR.

- the period is the number of values generated until the algorithm repeats itself.

**phase reset:** to restart a waveform at its initial value.

- for FM instruments, this restarts the volume envelope also.

**PSG:** programmable sound generator. any sound chip is a PSG, though the term is often used to specifically refer to chips that produce only simple waveforms and noise.

**pulse wave:** a waveform with a period consisting of only two amplitudes, high and low. also known as a rectangular wave.

**pulse width:** sometimes called *duty cycle*. in a pulse wave, this is the ratio of the high part to the high and low combined.

**release:** the part of a note that plays after it's no longer held, or the part of a macro the plays after it stops looping. usually applies at key off.

**resample:** to convert a sample to a different playback rate.

- this is a "lossy" process; it usually loses some amount of audio quality. the results can't be converted back into the original rate without further loss of quality.
- resampling to a lower rate reduces the amount of memory required, but strips away higher frequencies in the sound.
- resampling to a higher rate cannot recover missing frequencies and may add unwanted harmonics along with greater memory requirements.

**raw:** a sample or wavetable file without a header. when loading such a file, the format must be set properly or it will be a mess.

**register:** a memory location within a sound chip. "register view" shows all the relevant memory of all chips in use.

.s3m: ScreamTracker 3 Module.

**sample (1):** a digitally recorded sound. usually stored as some variant of PCM.

- these can take up a lot of room depending on length and sample rate, thus older systems tend to use short, lower quality samples.

**sample (2):** a single value taken from a digitally recorded sound. a sample(1) is made up of samples(2).

**signed:** a digital representation of a number that may be negative or positive.

- if an imported raw sample sounds recognizable but heavily distorted, it's likely to be unsigned interpreted as signed or vice-versa.

**software mixing:** mixing multiple channels of sound down to a single stream to be sent to a PCM channel.

- this puts a heavy load on the CPU of the host system, so it was rarely used in games.
- *Furnace*: this is used for DualPCM and QuadTone.

**square wave:** a wave consisting of only two values, high and low, with equal durations within the wave's period.

- this is equivalent to a pulse wave with a duty of 50%.

**supersaw:** a sound made up of multiple saw waves at slightly different frequencies to achieve a chorusing effect.

**tap:** a specified bit location within an LFSR.

**tick rate:** the number of times per second that the sound engine moves forward. all notes and effects are quantized to this rate.

- this usually corresponds to the frame rate the system uses for video, approximately 60 for NTSC and 50 for PAL.

**unsigned:** a digital representation of a number that can only be positive.

- if an imported raw sample sounds recognizable but heavily distorted, it's likely to be signed interpreted as unsigned or vice-versa.

.vgm: Video Game Music. a file containing the log of data sent to a sound chip during sound playback.

- saving to a .vgm file may be compared to "converting text to outlines" or similar irreversible processes. the results cannot be loaded back into the tracker.
- different versions of the VGM format have different capabilities, with trade-offs. older versions may lack chips or features; newer versions may not be compatible with some software.

- samples are stored uncompressed. PCM streams (such as DualPCM) can quickly take up a huge amount of space.

**waveform:** a very short period of repeating sound.

- the most basic waveform is a sine wave. others include triangle, pulse, saw, and the like.

**wavetable** (1): a very short looping sample.

**wavetable** (2): an ordered group of wavetables(1) used in sequence within a single instrument.

.xm: eXtended Module. the file format of songs made with FastTracker 2.

.zsm: ZSound Music. a VGM-like file meant specifically for the Commander X16 computer.



# hexadecimal

---

the hexadecimal numeral system differs from the decimal system by having 16 digits rather than 10:

HEX	DEC
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

when there is more than one digit, these are multiplied by 16, 256, 4096 and so on rather than 10, 100, 1000:

HEX	DEC
00	0
04	4
08	8
0F	15
10	16
11	17
12	18
13	19
20	32
30	48
40	64

HEX	DEC
### hex to decimal	

for example, take hexadecimal number AA:

$$\begin{array}{rcl}
 & \text{2nd digit} & \text{- \ /- 1st digit} \\
 & \text{A} & \text{A} \\
 16^1 \times 10 & = & 16 \times 10 = 160 + 10 = 170
 \end{array}$$

now for hexadecimal number 4C5F:

$$\begin{array}{rcl}
 & \text{3rd digit} & \text{- \ /- 2nd digit} \\
 \text{4th digit} & \text{- \ /- 1st digit} & \\
 4 & \text{C} & 5 \quad \text{F} \\
 | & | & | \quad | \\
 | & | & | \quad 15 = 15 = 15 + \\
 | & | & | \quad 16^1 \times 5 = 16 \times 5 = 80 \\
 | & | & | \quad 16^2 \times 12 = 256 \times 12 = 3072 \\
 | & | & | \quad 16^3 \times 4 = 4096 \times 4 = 16384 \\
 \hline
 & & = 19551
 \end{array}$$

## decimal to hex

---

if it's less than 16, just memorize the table at the top of this document.

otherwise find the power of 16 that is closest to the number you want to convert, but no larger than the number.

then divide, and take the remainder.

divide the remainder with the previous power of 16, until the divider is 1.

for example, for the decimal number 220:

$$\begin{array}{rcl}
 220 \div 16 & = & 13 \text{ (r = 12)} \quad \text{D} \\
 12 \div 1 & = & 12 \text{ (stop here)} \quad \text{C}
 \end{array}$$

= DC

now for decimal number 69420:

$$\begin{array}{rcl}
 69420 \div 65536 & = & 1 \text{ (r = 3884)} \quad 1 \\
 3884 \div 4096 & = & 0 \text{ (r = 3884)} \quad 0 \\
 3884 \div 256 & = & 15 \text{ (r = 44)} \quad \text{F} \\
 44 \div 16 & = & 2 \text{ (r = 12)} \quad 2 \\
 12 \div 1 & = & 12 \text{ (stop here)} \quad \text{C}
 \end{array}$$

= 10F2C

## bitmask

---

a bitmask is a value that actually represents a set of individual binary bits to be toggled, some of which may be grouped to form small binary numbers. these are used by adding up the value of each bit and converting the result to hex. in macros, these are shown in decimal.

BIT	BINARY	DECIMAL
bit 7	1000 0000	128

BIT	BINARY	DECIMAL
bit 6	0100 0000	64
bit 5	0010 0000	32
bit 4	0001 0000	16
bit 3	0000 1000	8
bit 2	0000 0100	4
bit 1	0000 0010	2
bit 0	0000 0001	1

for example, to turn bits 7 and 5 on, and put 110 (decimal 6) in bits 1 to 3:

BIT	BINARY	DECIMAL
bit 7	1... ..	128
bit 6	.0.. ..	0
bit 5	..1. ....	32
bit 4	...0 ....	0
bit 3	.... 1...	8
bit 2	.... .1..	4
bit 1	.... ..0.	0
bit 0	.... ...0	0

added together, the resulting binary is 1010 1100, which converts to hex as AC and to decimal as 172.

## hex to decimal table

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
60	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
80	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

# quick start guide

---

Furnace is amazingly versatile, but it can also be intimidating, even for those already familiar with trackers. this quick start guide will get you on the road to making the chiptunes of your dreams! if you're a beginner, it will probably take about an hour from start to finish.

this guide makes a few assumptions:

- you've already installed Furnace and know where to find the demo files that come with it. look for `quickstart.fur` but don't open it yet.
- you haven't changed any configuration or layout yet. it should start up with the default Sega Genesis system.
- you're working with a PC keyboard, US English, QWERTY layout. Mac users should already know the equivalents to the `Ctrl` and `Alt` keys.
- you're comfortable with keyboard shortcuts. if not, a lot of this can also be done using buttons or menus, but please try the keyboard first. it's worth it to smooth out the tracking workflow.

if an unfamiliar term comes up or you need more clarification on a term, refer to the [basic concepts<sup>9</sup>](#) and [glossary<sup>16</sup>](#) docs.

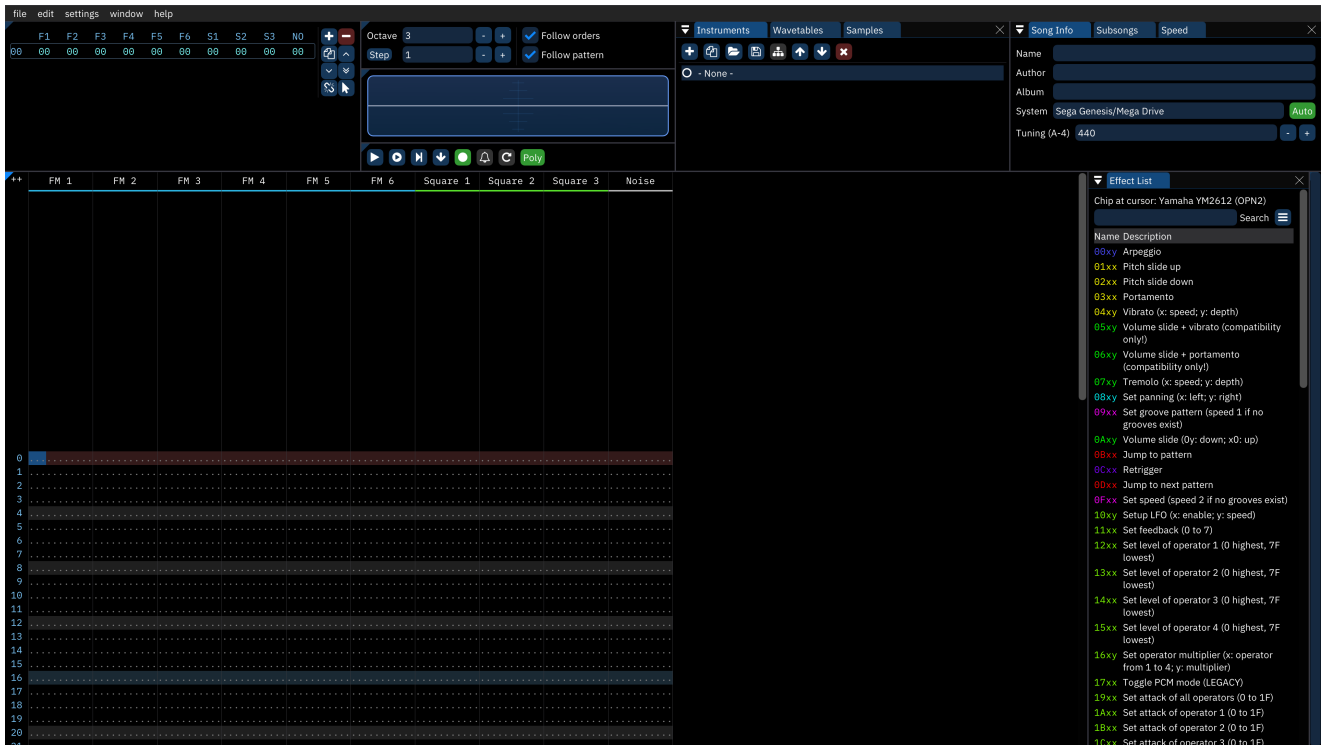
if you're not already familiar with the ImGui style of interface, you might want to take a quick glance at the [UI components<sup>51</sup>](#) documentation. if at any point something goes wrong with the interface – something gets moved to where it's inaccessible, something closes a window or tab unexpectedly, or the like – it can always be reverted to its original state by selecting "reset layout" from the "settings" menu.

with all that said, start up the program and let's get going!

## I've opened Furnace – now what?

---

on starting Furnace for the first time, the interface should look like this. if it's not quite right, drag the borders between sections until it approximately matches.



there's a lot going on, but the most prominent part of Furnace's interface is the **pattern view** – the spreadsheet-like table that takes up the bottom-left.

click to place the cursor somewhere in this view. it will appear as a medium-blue highlight. try moving around with the up and down arrow keys. also try the PgUp and PgDn keys to move around faster. the Home and End keys quickly move to the first and last rows. the vertical axis represents time. during editing and playback the view scrolls around a highlighted row that stays put in the center; this is called the **playhead**.

now try the left and right arrow keys to move between columns. pressing Home or End twice will shuttle you to the first or last column, respectively. when you're done, hit Home twice to return to the top-left.

let's play a little! notes are arranged on the keyboard rather like a piano. start with the bottom row of letters (ZXCVBNM). they should sound out the notes of the C major scale, like the piano's white keys. above that, we have the accidentals where the piano's black keys would be expected (SD GHJ). play with these, then move two rows up to find the same arrangement but one octave higher (white keys on QWERTYU, black keys on 23 567). these rows also extend a little further to the right into the next octave.

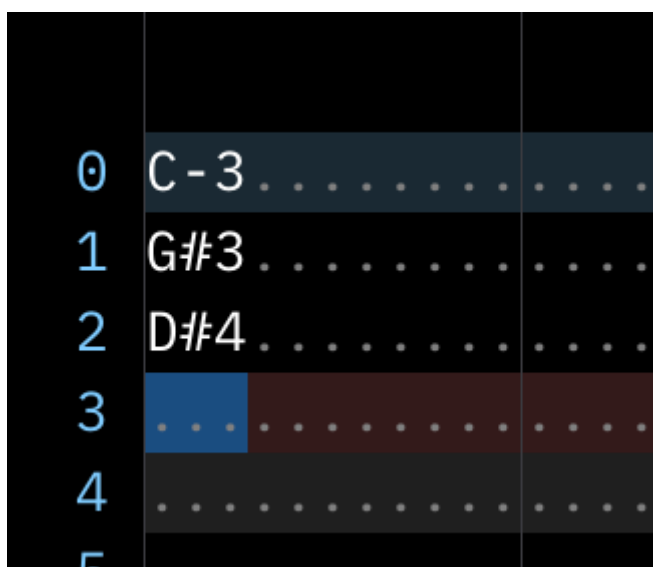


to change which octaves are represented on the keyboard, use the / and \* keys on the numeric pad. (if you don't have a numeric pad, these keys can be remapped; the [keyboard<sub>65</sub>](#) doc explains how.) as an alternative, there's an octave selector at the top of the interface, a third of the way in from the left.

now press the space bar to change from play to edit mode. the row the cursor is on will change to dark red – the playhead mentioned earlier. another way to tell what mode we're in is via the play/edit controls just above the pattern view in the center; make sure the "record" button is on.



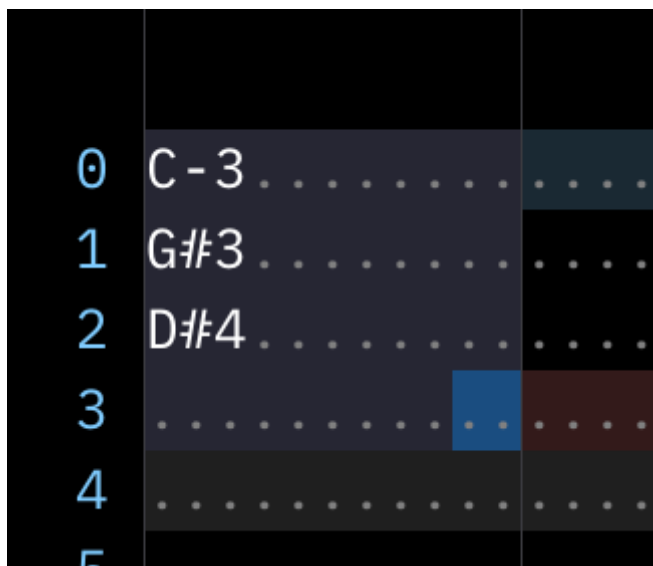
now try playing some notes; they should appear in the pattern view, one after another.



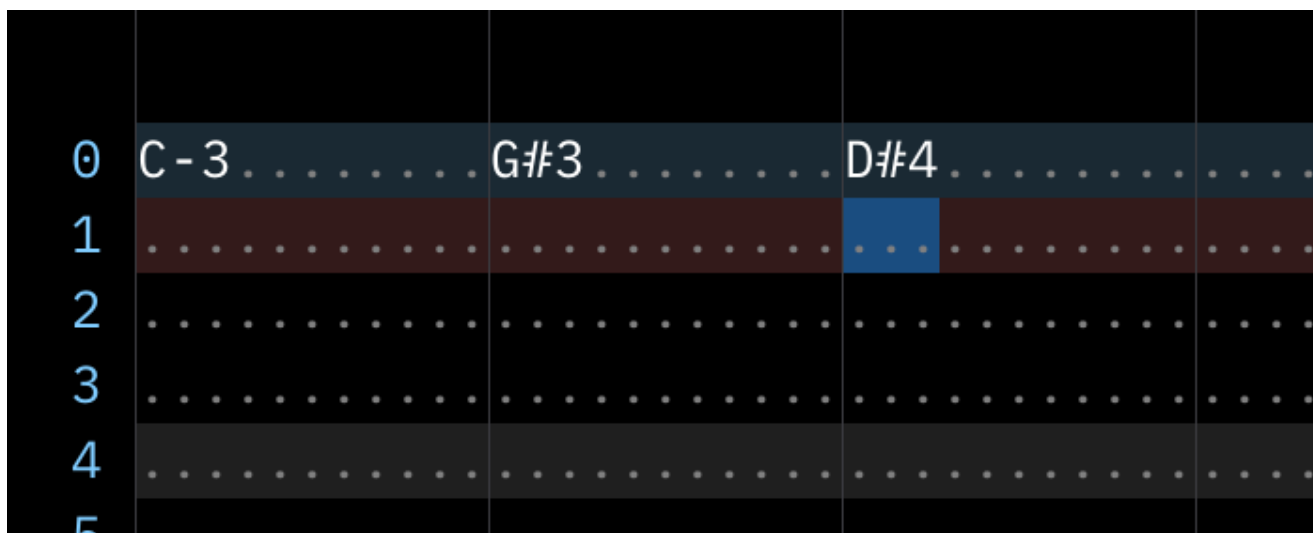
each **channel** is a group of columns separated from the others by lines, with a name at the top. each channel can only ever play one note at a time. to hear this in action, move the cursor back to the top and press the Enter or Return key to start playback. you should hear the notes you

entered played back quickly, one after another, each cutting off the previous note. if you let it play long enough, it'll wrap around to the start to go through them again; press Enter again to stop playback.

now let's clear out those notes. you could delete them individually with the Del key, but let's try something else first. click and drag to select them all. you'll know they're selected when they have a medium grey background. hit Del to delete them all at once.



you'll usually want more than one note playing at a time. move back to the start of the pattern in the leftmost column of the leftmost channel; this should clear the selection area. put some different notes next to each other in the same row. only enter notes in the first column of each channel; we'll get to those other columns later. (don't do more than six notes at once yet. we want to stay in the channels labelled "FM" for now.) once those are in place, go back to the top row and use the Enter key to start playback. they should all sound at the same time as one single chord.



that chord will ring out for quite some time, but let's try stopping it early. a couple rows after that chord, use the Tab or 1 key to enter a **note off** (sometimes called "note cut") in each channel that has a note. it'll appear as OFF in the note column. now try the shortcut F5 to play from the start without having to move there. you should hear the chord as before, then it will stop where the note offs are, as though letting off the keys of a piano.



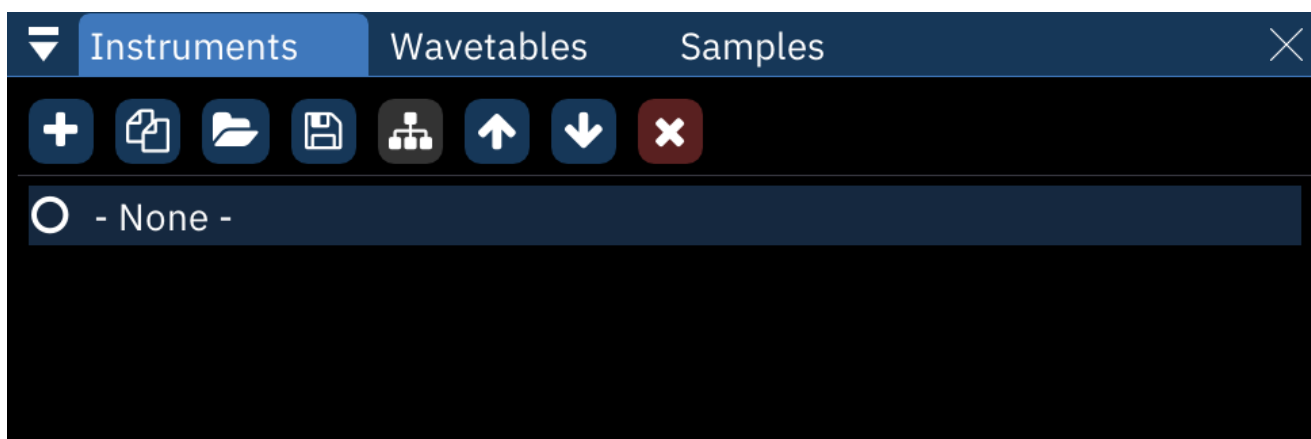
0	C-3 . . . . .	G#3 . . . . .	D#4 . . . . .	
1	. . . . .	. . . . .	. . . . .	. . . . .
2	OFF . . . . .	OFF . . . . .	OFF . . . . .	. . . . .
3	. . . . .	. . . . .	. . . . .	. . . . .
4	. . . . .	. . . . .	. . . . .	. . . . .
5	. . . . .	. . . . .	. . . . .	. . . . .

of course, errors can happen. let's pretend those note offs were a bad idea and undo them with `Ctrl-Z`. Furnace keeps track of multiple levels of undo. undo will work for the pattern view, most text entry boxes, and a few other places; try it out here and there along the way to get a sense for what it can undo for you! for now, let's change our minds again and put those note offs back with redo, which is `Ctrl-Y`.

before the next part of this guide, save the current **module** – the tracker file that contains everything needed for a song. use `Ctrl-S` and pick a good spot on your computer for the file. Furnace modules always have a filename that ends in a `.fur` extension.

## how do I get different sounds?

at the top of the interface, just right of center, is the **instrument** list. there are also tabs for wavetables and samples, but we'll get to those later. just like physical instruments, these define the sounds we can use in the track. unlike physical instruments, these sounds can be endlessly redefined.

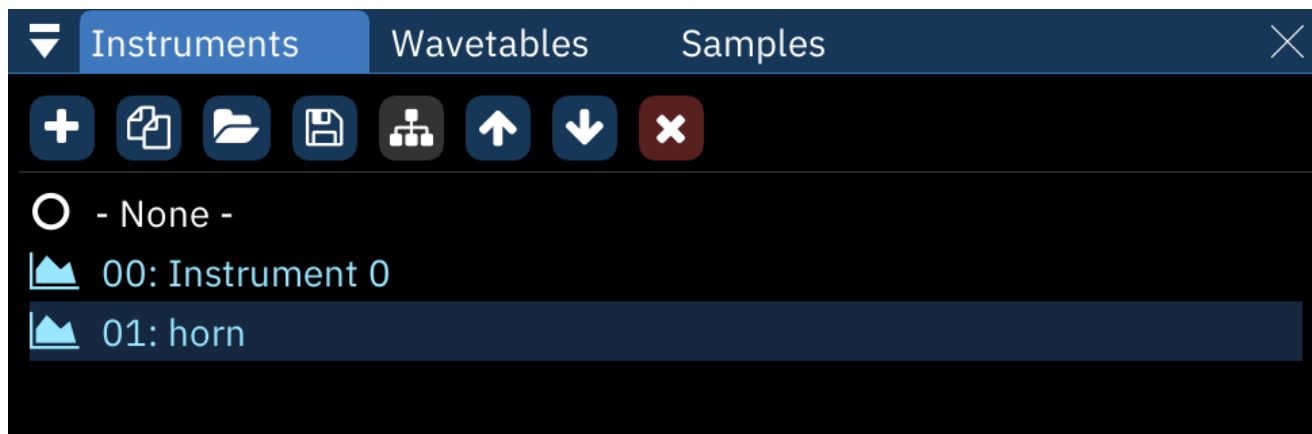


click the `+` button to add a new instrument. a list of instrument types will pop up, one for each type supported by the chips in use. select "FM (OPN)", and the new instrument will appear in the list as "00: Instrument 0". this will sound the same as the default instrument (listed as "- None -").

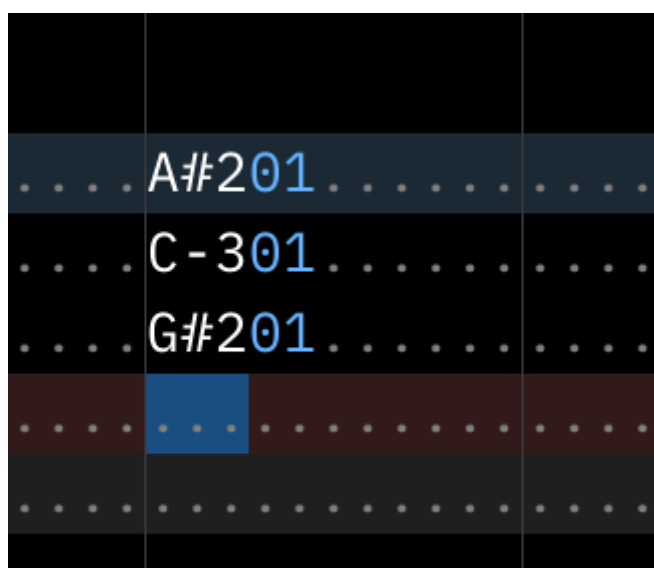
we still need something new and different, so let's pull from another module. open up a second instance of Furnace and use `Ctrl-O` to open the `quickstart.fur` file included with Furnace in its

demos directory. the instrument list will contain "00: horn"; select it, then use the floppy-disk save icon above it to save it wherever you like. Furnace instrument filenames end with the .fui extension.

let's return to the first instance with our slowly-evolving practice track. load up the new instrument; click the folder button left of the "save instrument" button and select the file. it will appear in the list as "01: horn", and it should already be highlighted.



click into the pattern view and add some notes in another FM channel well after our existing chord. we'll hear them with our new sound, and the number 01 appears next to them. this is the instrument column, and it can be edited directly by typing in the number desired. generally, each note should have an associated instrument value.



## how do I change volume?

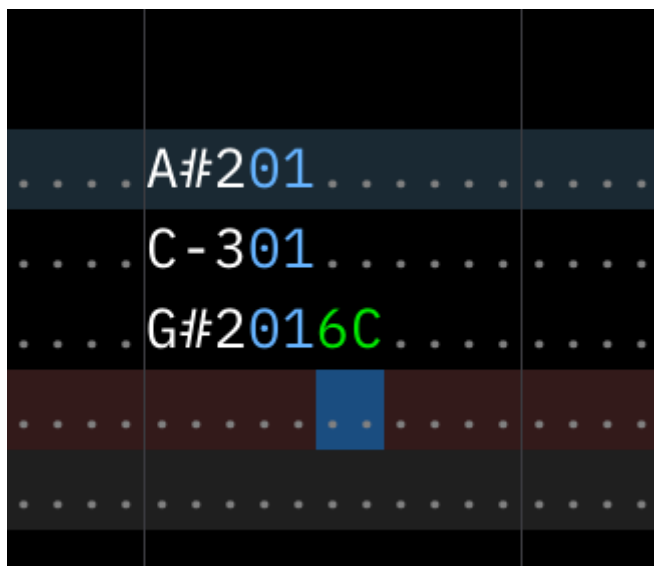
next to the instrument column is the **volume** column. typing in it will change the loudness of the associated note... but it's not always as straightforward as it seems.

for one thing, this column operates in [hexadecimal](#)<sup>21</sup>. in fact, so does the instrument column, and so will the others when we get to them. if you're ever uncertain what the decimal equivalent is, put your cursor over the volume in question and look to the menu bar, which doubles as a status bar.

after the menus, it will show "Set volume:", then the decimal value, hexadecimal value, and percentage of full volume.

also, if you haven't saved your recent edits, there will be an indicator at the end which shows the "modified status". it might be worth saving now.

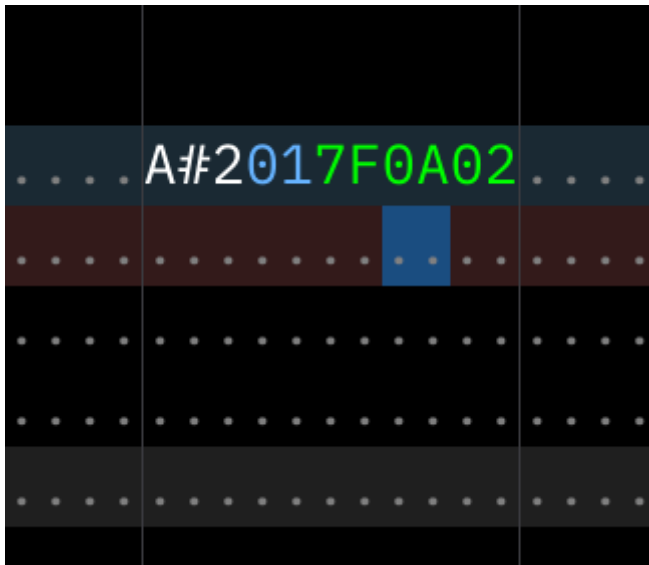
try typing 6C into the volume column of the last note, then play it back. it should be much quieter than those before it in the column because they default to full volume – in this case, 7F. everything in the column after our 6C will inherit that volume level until it's changed again, so you don't have to enter volumes for every single note.



now try putting 90 in the volume column. it automatically changes to 7F because that's the maximum volume available for this channel. different channels may have different maximum volumes because of how each chip works.

it gets a little stranger yet. some chips use "linear" volume, which translates directly to amplitude; dividing the value in half results in half the volume. other chips, such as the one we're using right now, use "logarithmic" volume; subtracting from it lowers the volume based on how loud it *sounds*. in this particular case, subtracting 8 to get 77 lowers the volume by half; subtracting 8 again to get 6F lowers it to one-quarter. this takes some getting used to, but it's more convenient in some ways.

there are more ways to change volume using **effects**. clear out all of our notes and place one at the top of one of the FM channels, and set it to full volume (7F). next to the volume column are the effect columns. the first of them is the effect type column, and it stores... the type of effect. in that column, type 0A; this corresponds to the "volume slide" effect. next to that, in the effect value column, type 02. hit the F5 key to play from the start, and you'll hear the note play, but instead of staying at a steady volume it'll smoothly fade out.



0A is an interesting effect because its two-digit value is split. look to the "effect list" to the right of the pattern view and find 0Axy. the description includes "(0y: down; x0: up)". this means that while value 01 is a slow fade out and 0F is the fastest, a value of 10 is a slow fade in and F0 is the fastest. try it now; go to row 10 in the same channel and type an effect of 0A20, then play again. the note will fade out until it hits that new effect, then fade back in to full volume! also note that putting the cursor on an effect will show the effect type and description in the status bar.

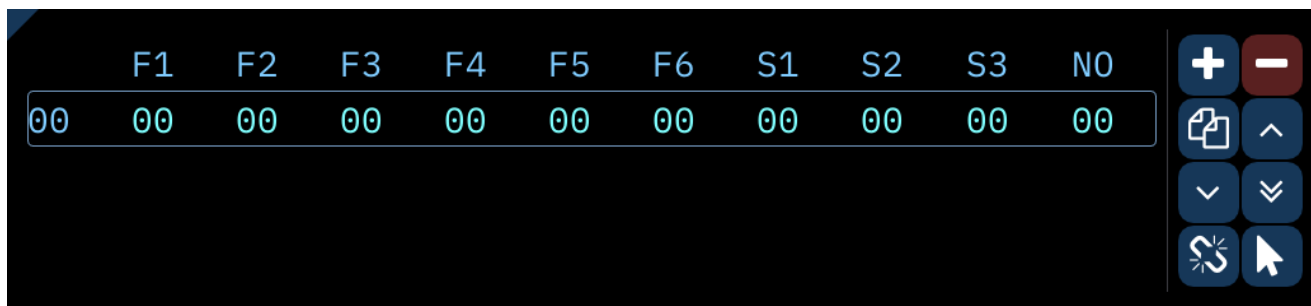
it's important to know that *most* effects are continuous, meaning they will continue to do what they do until explicitly stopped. volume slides are like this. place an effect type of 0A on row 16. you can leave the effect value blank or type 00 there; these are equivalent, and both will stop the effect. play from the start, and when the note fades back in, it'll stop short of full volume and remain there.

common effects are explained more thoroughly in the [effects<sub>104</sub>](#) documentation. each chip may have its own specialized effects, which are covered in the [systems](#) docs. however, those are best explored later.

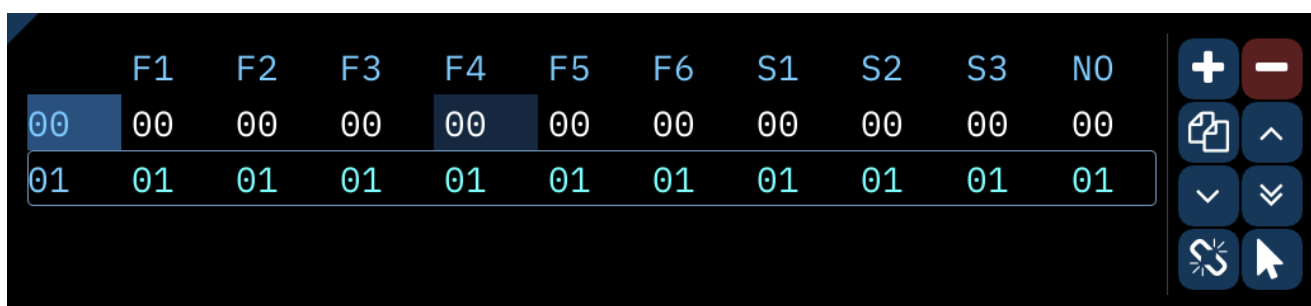
on row 20, add a different note without a volume. play from the start, and you'll hear that the new note plays at the volume the previous one left off at. the result of the volume slide is kept in the "memory" of the channel. enter a volume of 7F and play again; it will start at full volume, then ramp down because the 0A volume slide is still going.

## how do I make the song longer?

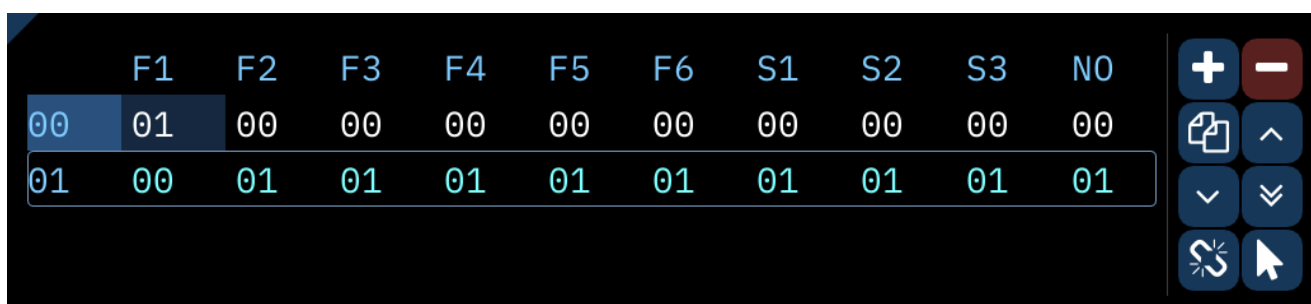
right now, our track is only about six and a half seconds long. this is because we only have one **order**. see, the term "pattern view" is slightly misleading in that a **pattern** is just one channel's worth of data; the pattern view shows all the patterns in an order at once. this can get confusing because sometimes both terms are both used to mean what we call an **order**, sometimes even within Fur nace itself.



at the top left of the interface we find the order view. similar to the pattern view, it's like a spreadsheet, but even simpler. from left to right, the top line shows short names for all the channels. each row of numbers beneath that shows which patterns play in that order. for the moment, only the first order 00 appears. click on the + button to the right of the row of channel labels, and another order row appears, not only labeled 01 but filled with that same number. click in the pattern view and move to the top-left by hitting Home twice. you'll see that the new patterns are empty. the pattern view shows the end of the previous pattern but faded out. try moving between these by clicking on their order numbers in the order list.



go to the first order and make sure there are some notes in the first channel. now click on the pattern number to the right of the order number (in the "F1" column); it will increase to 01, and the notes we could see in the pattern view have disappeared! not to worry, they're still stored in the that channel's pattern 00. select order 01 and right-click that first pattern number; it will decrease to 00, and the notes in it will reappear. this way, you can rearrange and reuse parts of your track without having to duplicate them all the time.

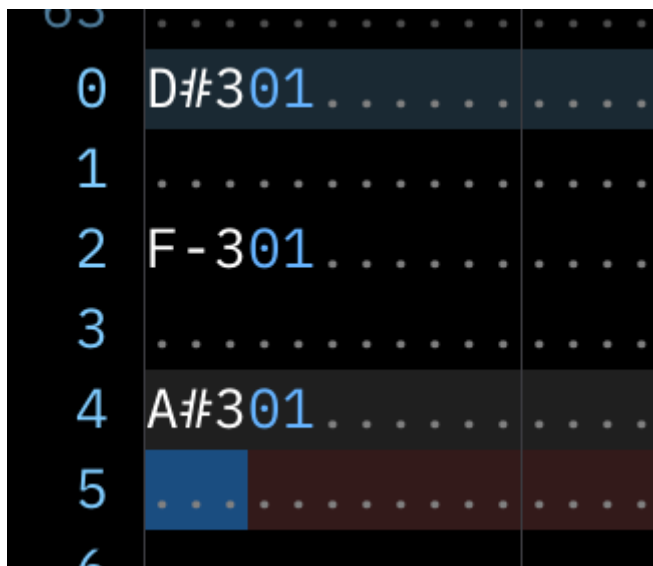


go back to the first order and put some notes in the first channel's pattern. in the order view, right-click the button showing two overlaid pages; this is the "duplicate" button. clicking it normally will add a row that repeats the current order. right-clicking that button creates a "deep clone", meaning that all the patterns in it are duplicated to new pattern numbers. when you want to make variations of the same patterns, this is somewhat faster than cut-and-paste (Ctrl-X and Ctrl-V, by the way, with Ctrl-C for copy).

the important take away here is that patterns exist independently of orders. the order list is a playlist of patterns that can be freely rearranged.

# how do I change tempo?

**tempo** and **speed** are a little tricky – in fact, for the purposes of Furnace, they mean different things! first, let's clear out our first order and put some evenly-spaced notes there.



the most basic unit of time is the **tick**. almost always, videogame systems take actions based on each frame of video, and these most often happen at 60 times per second, usually expressed as 60Hz. (this is for NTSC systems; systems that expect PAL will use 50Hz, and arcade games can use all sorts of different values...) because of this timing, everything that happens during playback will happen on a tick, never in between ticks.

if we click on the "Speed" tab at the top-right of the interface, we'll see the "Base Tempo" line at the top shows the tick rate as "60Hz" to the right. we could change the base tempo to something arbitrary and the tick rate would change accordingly, but this wouldn't be authentic to the system's capabilities, so let's leave the base tempo at 150. we see the calculated tempo three lines down, to the right of the input for "Divider"; it reads "150.00 BPM".



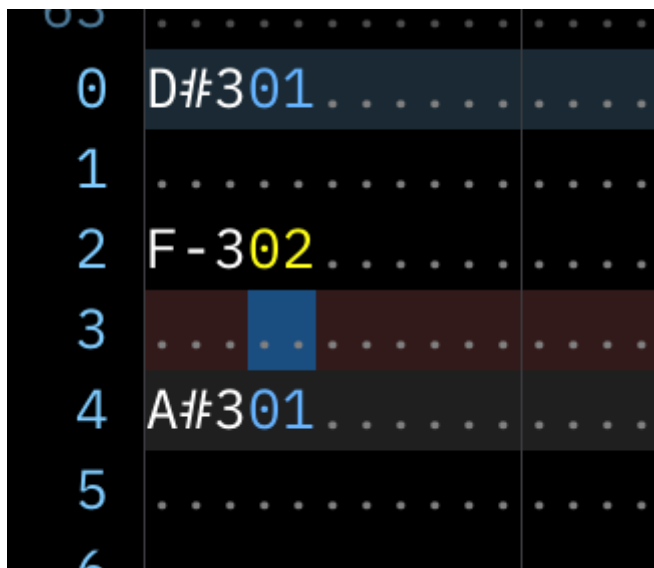
beneath Base Tempo is "Speed", set to 6. right now, each row takes 6 ticks to complete before moving to the next row. let's say we want things to be a little faster. play the current set of notes to hear their tempo first. then, change speed to 5; the tempo after "Divider" will now show "180.00 BPM". play our notes back, and they're definitely faster... perhaps faster than desired. it's possible to get tempos in between by alternating speeds; if you're interested, check out the documentation on [speeds and grooves](#)<sup>372</sup> later on.

## what about those other channels?

---

here's where we really get into the nitty-gritty of our emulated videogame system. we've been using Furnace's default system, the Sega Genesis. it employs two very different sound chips. the first is the Yamaha YM2612, also known as the Yamaha OPN2; it uses frequency modulation (FM) synthesis to generate sounds, and that's what we've heard so far. the other sound chip is the Sega PSG; it's a programmable sound generator (PSG) that can only make square waves and variations of noise. it's nowhere near as versatile, but don't ignore it – it's an important part of the classic Genesis sound.

let's start by creating a new instrument, this time choosing "SN76489/Sega PSG" from the list. the new "Instrument 2" appears in the instrument list, already selected. now click in the pattern view and change one of our existing notes to use the new instrument. the number will change color from soft blue to bright yellow; this means that the chosen instrument isn't meant for the chip it's being used on, and if played back, we'll only hear that familiar default FM instrument.



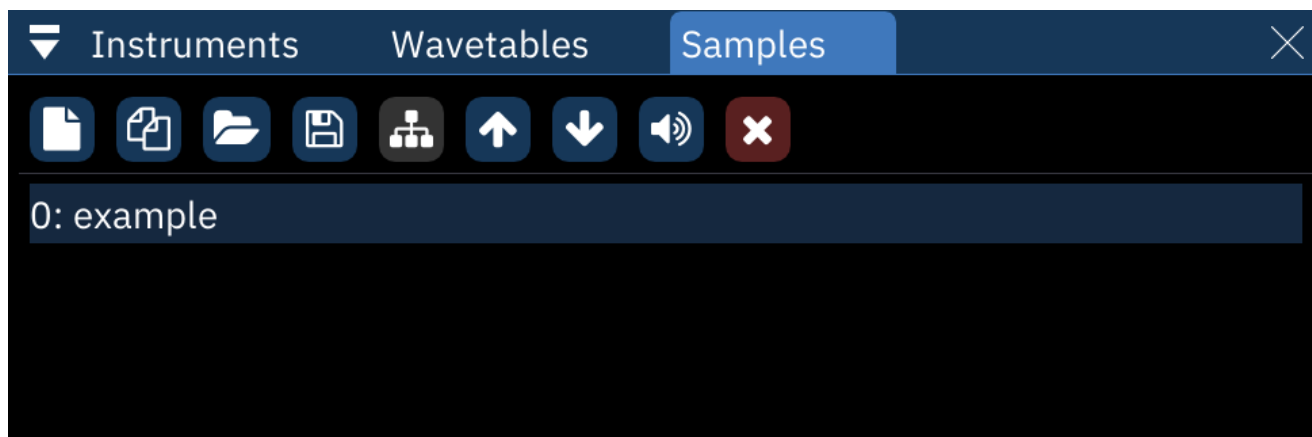
go ahead and undo that edit, then move to the channel labelled "Square 1", the first of the PSG's channels. try adding notes with the new instrument, and they'll work just fine without complaint. of course, they're plain, no-frills square waves. while we're here, try making them quieter by entering new volumes; since this chip only uses sixteen volume levels, 0F is the maximum.

let's move to the noise channel now. the same instrument will work here, but playing different notes gets us different "pitches" of noise. this channel is both more and less versatile than it seems, with several notable quirks that we won't get into here, but take a look at [this chip's documentation](#)<sup>298</sup> later on.

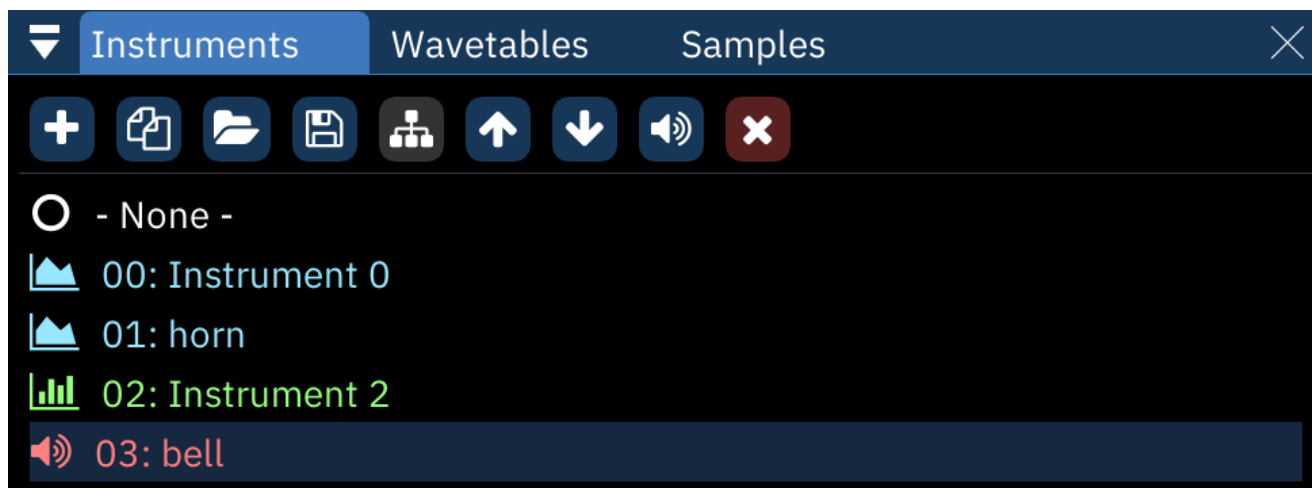
## what about samples?

the FM side of the Sega Genesis has a special feature; channel 6 can be used to play back digital **samples**. this means that any recording – a snare drum, an orchestra hit, somebody talking, whatever you have – can be part of the music.

go back to that second instance of Furnace. just as we saved an instrument last time, let's switch to the "Samples" tab and select the lone sample there, "0: example", then save it as a .wav file. swap back to the instance of Furnace we've been working in, and in the Sample tab, open it.



in order to use the sample, we want to make an instrument that references it. right-click on it in the list and select "make instrument". the "Instrument Editor" window will pop up to show us that we now have an instrument 3 named "example", a type of "Generic Sample", and below that, the sample selected is "example". while we're at it, let's change the instrument name to "bell" since that's what it sounds like; just select the "example" at the top and type over it.



now, let's hear it in action. close the instrument editor, then clear out everything in the patterns of our first order. (either delete what's there, or adjust orders to get it out of the way. try hitting Ctrl-A three times to select everything at once.) switch to our new bell instrument and put a C-4 note in channel "FM 6". when we play it back, it sounds perfect!

if ever a sample sounds out of tune, refer to the [sample tuning guide](#)<sup>398</sup> to fix it up.



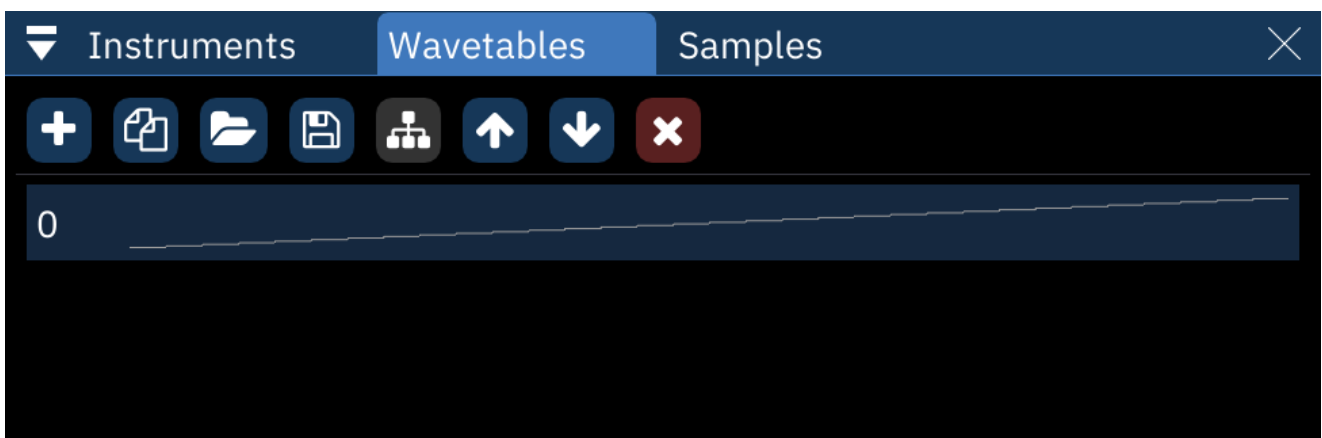
an important note: in this case, we can use a Generic Sample instrument type just fine, but there are chips that use samples in specialized ways. always check [the chip's documentation](#) for the best way to use samples with it.

now that we've gotten everything we need from `quickstart.fur`, close that instance of Furnace.

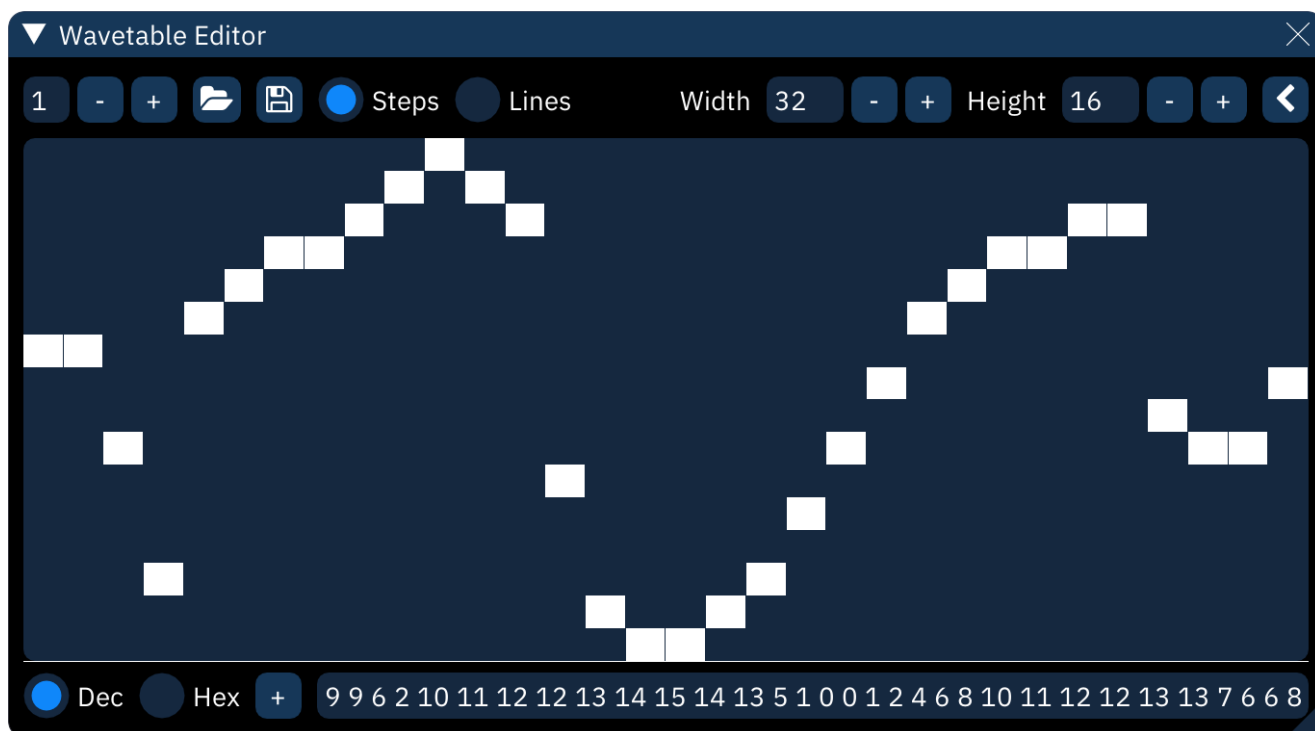
## what about wavetables?

some chips can use **wavetables**, which are a lot like very short looping samples. one of these is the Game Boy. let's start a new file, either from the menu or with `Ctrl-N`. a warning dialog asks if you want to save; it's up to you. after that, a dialog box pops up to ask which system we want; type "boy" in the search and it will be at the top of the results. select it.

in our brand new song, we'll want to add a new wavetable. between the "Instruments" and "Samples" tabs, select "Wavetables". add a new one with the + button. Furnace will generate a wavetable of the right size for the current chip, and it will already have a sawtooth wave in it.



double-click that new entry to open the "Wavetable Editor". you'll see a line of large pixel-like blocks. this is our sawtooth wave. nice as those are, let's get creative. click anywhere in that area and "draw" a new wave, something interesting. note that at the bottom of the window, there's a line of numbers that change as you edit. you can edit the numbers directly to change the values above; keep this in mind for later. once you're happy with this wavetable, make at least two more, just for later demonstration purposes.



right now, we can't do much with this wavetable; as with samples, it needs an instrument. this time we can't just create one directly from the wavetable, so go to the instruments list and add a new one. open it up in the instrument editor, name it whatever you like, then select its "Macros" tab.

we'll get to macros in more detail in a bit, but for now, simply click the down-arrow next to "Wave form". click the + button that appears; a column will turn grey in the box to the right. click in the middle of that column and it turns half orange; this is how you select which wavetable to use for this instrument. it's like a bar graph.



close the wavetable editor, move the instrument editor off to the side, and click into the "Wavetable" channel in the pattern view. add a few notes to get a good sense of their tone. in the instrument editor, change that macro to a different value, and play the notes again to hear the difference between the wavetables you made.

## but... what's a macro?

---

the **macro** is perhaps Furnace's most powerful feature. formally defined, it automates a note's parameters while it plays. a lot of what can be achieved with effects can be done with macros, but on a per-tick basis instead of per-row.

let's start by clearing out the notes we've entered. after that, move the cursor to the top-left into the "Pulse 1" channel. create a new instrument and go into the instrument editor. we'll want to work with the volume macro, but before we can do that, we have to select the "Game Boy" tab and check the box labelled "Use software envelope". the Game Boy's sound hardware can do its own limited volume envelopes, but those won't help us right now, and if we leave the box unchecked, the volume macro won't work (though the others will).

in the "Macros" tab, the number input field that appears beneath the word "Volume" is the length of the volume macro; let's set it to 30. in the large box next to it, draw a ramp from near minimum volume (1) to maximum volume (15) at the left, then another down to minimum volume (0) at the right. if it's a little uneven, that's okay; you can always edit the numbers directly beneath the box, just as with the wavetable editor. also try right-clicking in the macro and dragging; now you have perfectly smooth ramps.

you may have a little trouble navigating the whole macro at once. use the scrollbar at the top of the macros tab to move around it. even better, use the - button to the left of it to narrow the bars until it's all visible!



while in the instrument editor (and as long as you're not in a text box) you can play notes on the keyboard without affecting anything in the pattern view. give it a try, and you'll notice that when held down, each note does its own quick fade in then fade to silence. you could do this with effects, but if you're using it on many notes, doing it with the instrument itself could save a lot of typing!

in the pattern view, add a few notes spaced far enough apart that the whole rise and fall is audible (at speed 6, five rows will do). then look to the thin bar underneath the macro view. it may not look like much, but if you hold the Shift key and click directly underneath the peak of the macro, it will light up green. we've just set a **release point**. play with the instrument a little here, and notice that

holding the key down holds the note in place at top volume – at the release point – until let go. now play the song from the start; each note will rise to max volume then stay there until the next note plays.



about ten rows after the last note in our song, place a note off. the final note rises to maximum, then is suddenly cut off! to get the rest of the macro to play, move your cursor over the note off and use the ~ key to replace it with a **macro release** instead, which will appear as REL. now when the song is played back, the final note will rise and hold steady until it reaches the macro release, then we'll hear the rest of the macro play out.

macros are absurdly powerful tools. read the [macro documentation](#)<sup>112</sup> to make the most of them!

## what's next?

---

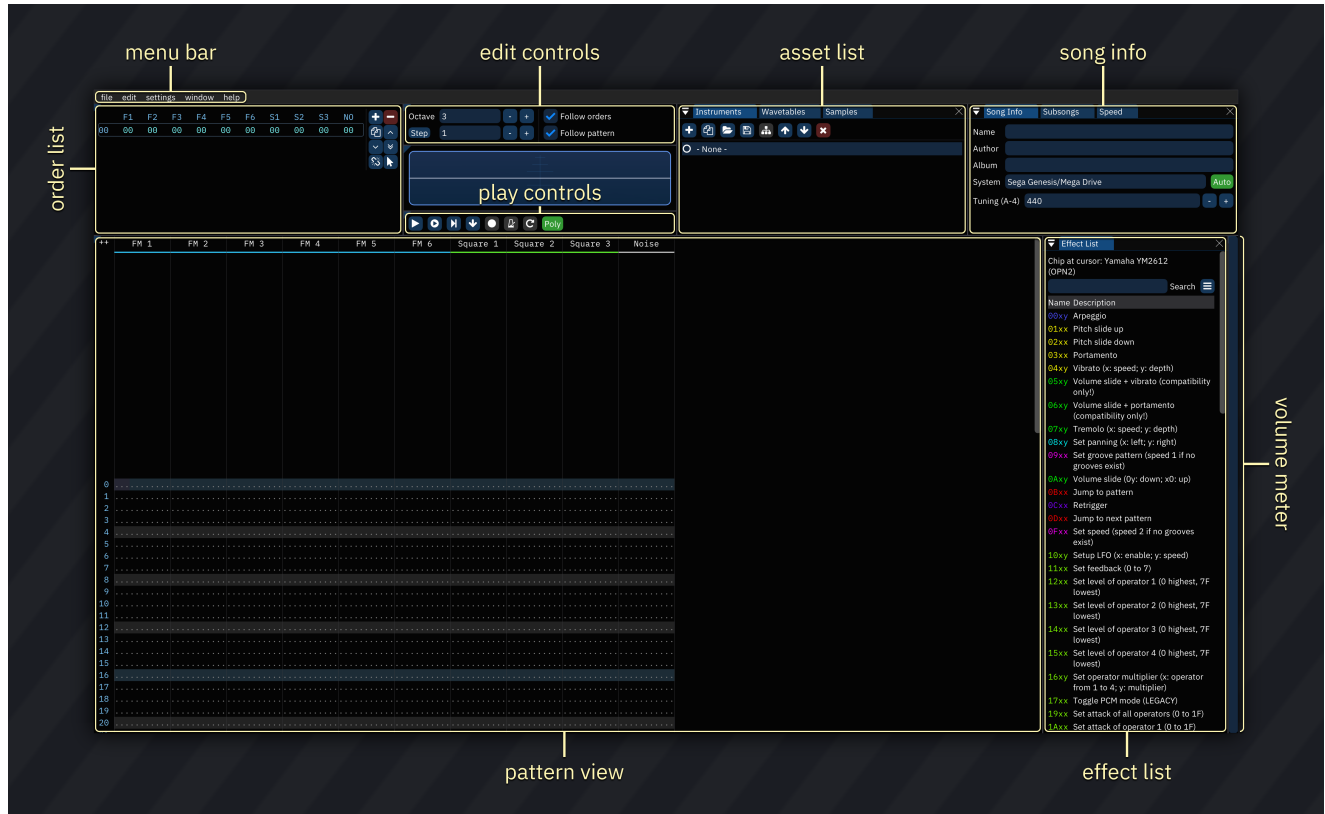
now you know the basics of how to make music with Furnace. from here, the rest of the documentation should make more sense, and it should be your primary reference. if you have questions that aren't answered there, feel free to ask in the [Discussions section](https://github.com/tildearrow/furnace/discussions) (<https://github.com/tildearrow/furnace/discussions>) on Furnace's GitHub repository.

most of all, don't be afraid to experiment. go play!

# interface

the Furnace user interface is where the job gets done.

the default layout of Furnace is depicted below.



## general info

- UI components<sup>51</sup>: read first!
- global keyboard shortcuts<sup>65</sup>
- menu bar<sup>72</sup>

## primary windows

- orders<sup>76</sup>
- play/edit controls<sup>78</sup>
- instrument/wavetable/sample list<sup>46</sup>
- song information<sup>97</sup>
- pattern view<sup>99</sup>
- effect list window<sup>57</sup>
- instrument editor<sup>112</sup>
- wavetable editor<sup>211</sup>
- sample editor<sup>216</sup>



## advanced topics

---

- [song comments](#)<sup>366</sup>
- [channels](#)<sup>354</sup>
- [chip manager](#)<sup>360</sup>
- [pattern manager](#)<sup>382</sup>
- [mixer](#)<sup>378</sup>
- [compatibility flags](#)<sup>367</sup>
- [oscilloscope](#)<sup>381</sup>
- [oscilloscope \(per channel\)](#)<sup>356</sup>
- [oscilloscope \(X-Y\)](#)<sup>390</sup>
- [clock](#)<sup>362</sup>
- [grooves](#)<sup>372</sup>
- [log viewer](#)<sup>376</sup>
- [register view](#)<sup>386</sup>
- [statistics](#)<sup>387</sup>
- [memory composition](#)<sup>377</sup>

## other topics

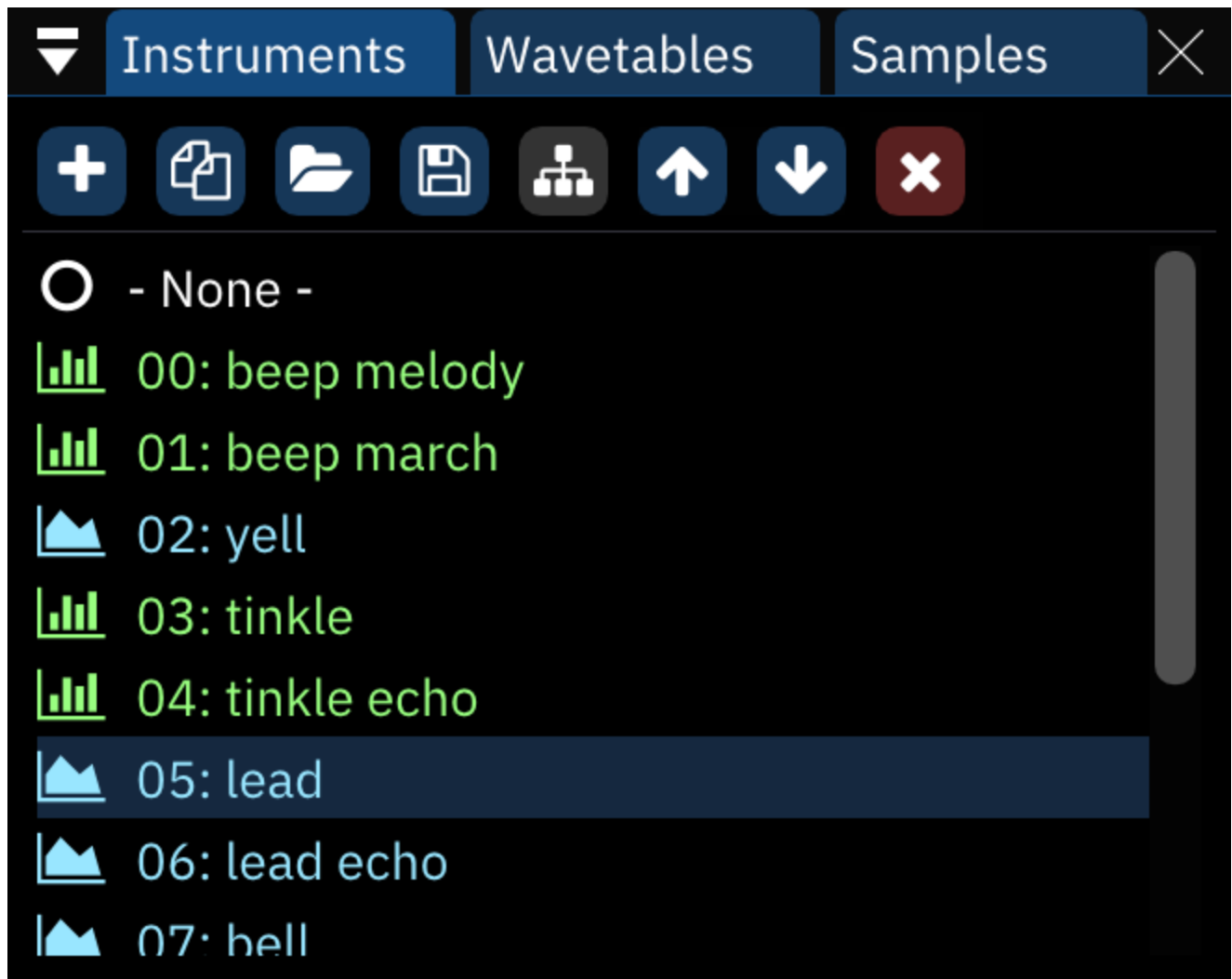
---

- [piano/input pad](#)<sup>384</sup>
- [settings](#)<sup>82</sup>

# asset list

an "asset" refers to an instrument, wavetable or sample.

## instrument list



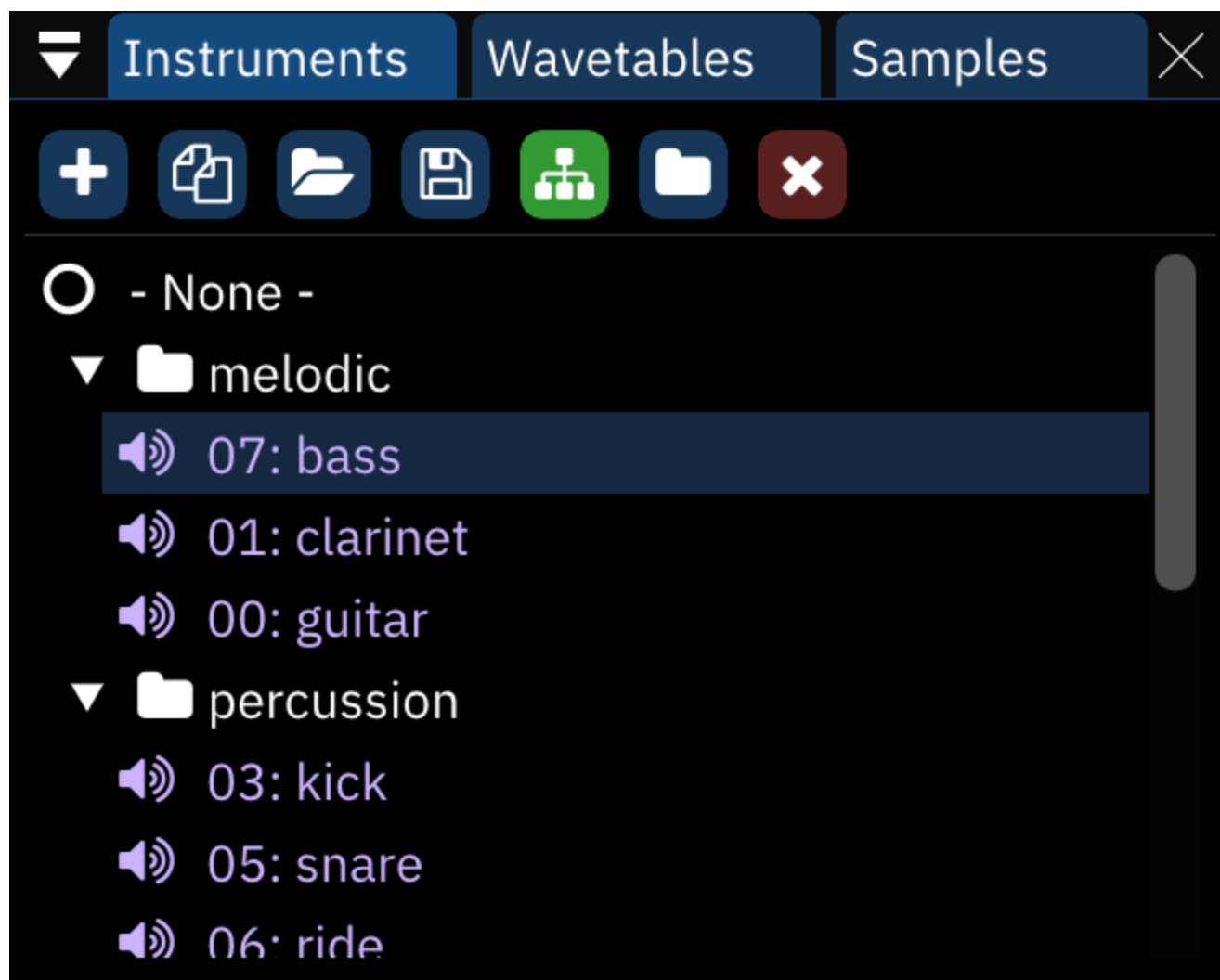
buttons from left to right:

- **Add**: pops up a menu to select which type of instrument to add. if only one instrument type is available, the menu is skipped.
- if the "Display instrument type menu when adding instrument" setting is disabled, this skips the menu and creates an instrument according to the chip under the cursor.
- right-clicking always brings up the menu.
- **Duplicate**: duplicates the currently selected instrument.
- **Open**: brings up a file dialog to load a file as a new instrument at the end of the list.
- if the file is an instrument bank, a dialog will appear to select which instruments to load.
- **Save**: brings up a file dialog to save the currently selected instrument.
- instruments are saved as Furnace instrument (.fui) files.

- right-clicking brings up a menu with the following options:
  - **save instrument as .dmp...**: saves the selected instrument in DefleMask format.
  - **save all instruments...**: saves all instruments to the selected folder as .fui files.
  - **Toggle folders/standard view**: enables (and disables) folder view, explained below.
  - **Move up**: moves the currently selected instrument up in the list. pattern data will automatically be adjusted to match.
  - **Move down**: same, but downward.
  - **Delete**: deletes the currently selected instrument. pattern data will be adjusted to use the next available instrument in the list.

instruments may be dragged and dropped to reorder them. this will change instrument numbers throughout the module accordingly.

## folder view



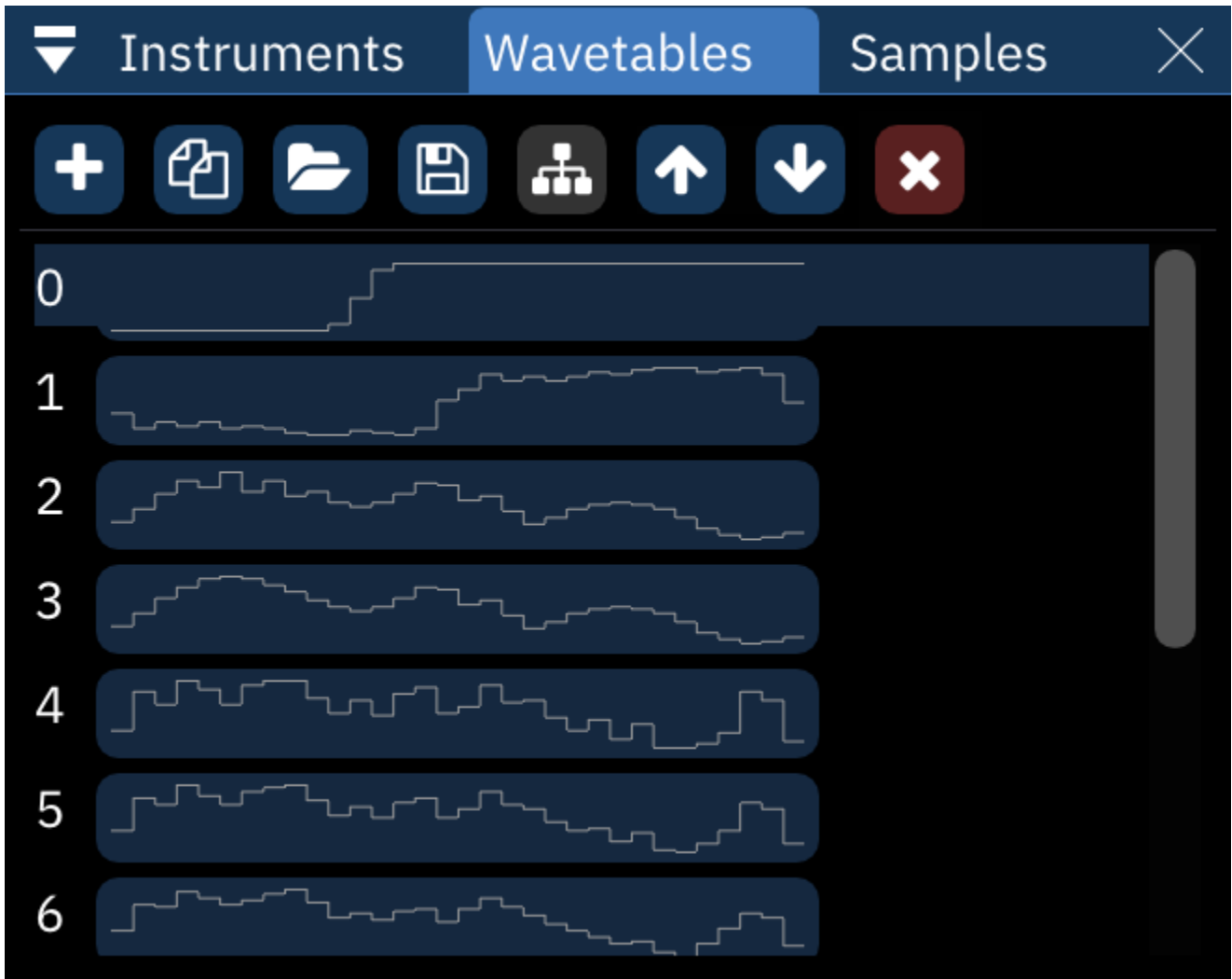
in folder view, the "Move up" and "Move down" buttons disappear and a new one appears:

- **New folder**: creates a new folder.

assets may be dragged from folder to folder and even rearranged within folders without changing their associated numbers.

right-clicking on a folder allows one to rename or delete it. deleting a folder does not remove the instruments in it.

## wavetable list



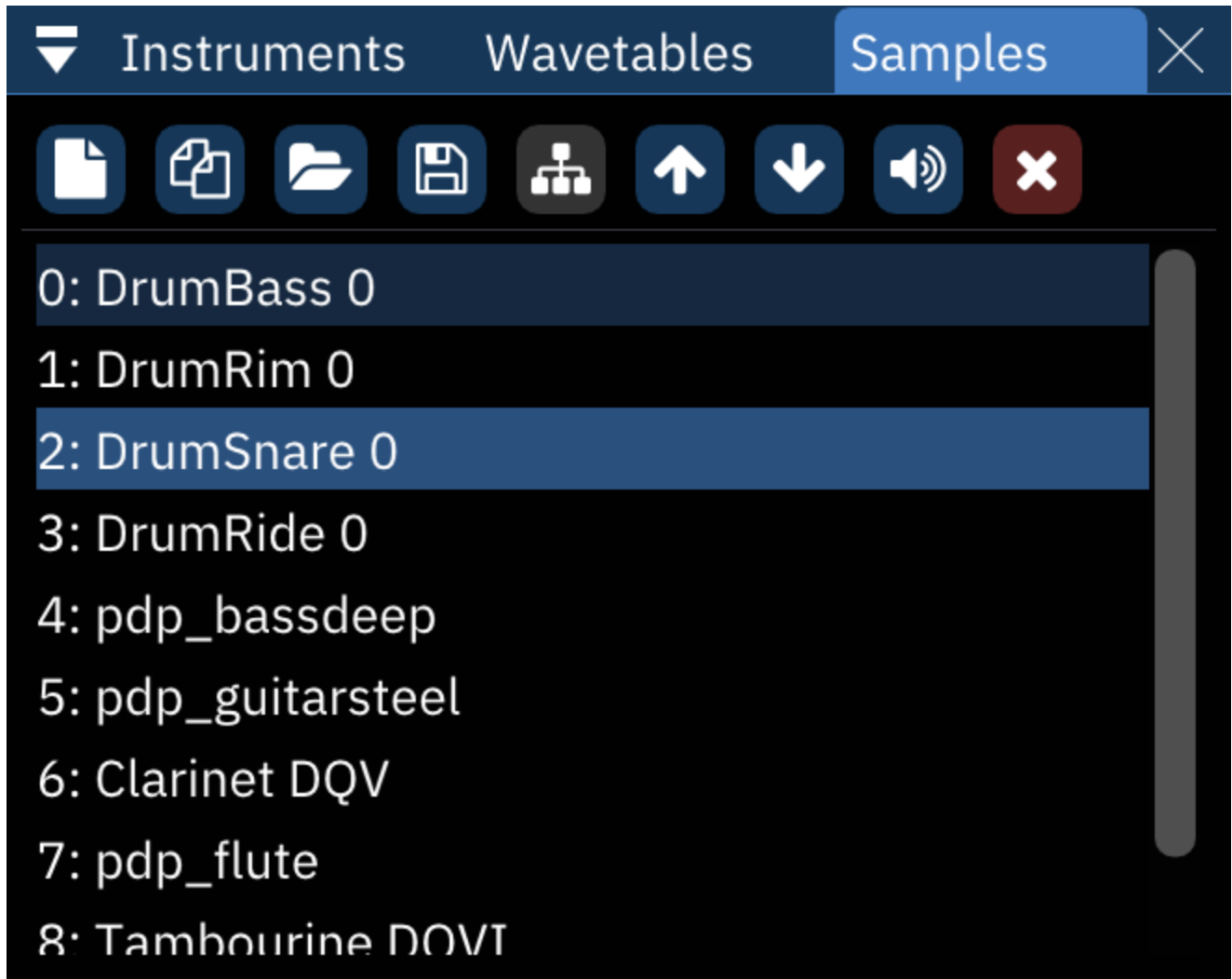
everything from the instrument list applies here also, with one major difference: moving waves around with the buttons or dragging them will change their associated numbers in the list but **not** in pattern or instrument data. be careful!

wavetables are saved as Furnace wavetable (.fuw) files.

right-clicking the Save button brings up a menu with the following options:

- **save wavetable as .dmw....**: saves the selected wavetable in DefleMask format.
- **save raw wavetable....**: saves the selected wavetable as raw data.
- **save all wavetables....**: saves all wavetables to the selected folder as .fuw files.

## sample list



everything from the wavetables list applies here also, with the addition of one button before the Delete button:

- **Preview:** plays the selected sample at its default note.
- right-clicking stops the sample playback.

samples are saved as standard wave (.wav) files.

right-clicking the Save button brings up a menu with the following options:

- **save raw sample...:** saves the selected sample as raw data.
- **save all samples...:** saves all samples to the selected folder as .wav files.

right-clicking a sample in the list brings up a menu:

- **make instrument:** creates a new instrument which is set to use the selected sample.
- **make me a drum kit:** allows you to instantly create a drum kit using all the samples in the list. see the next section for more information.
- **duplicate:** makes a copy of the selected sample.
- **replace...:** opens a file dialog to choose a replacement sample.

- **save:** opens a file dialog to choose where to save the sample.
- **delete:** removes the sample.

## make me a drum kit

I have added this option to make it easier for you to create a drum kit.  
it puts all the samples into a new instrument with sample map.

after selecting this option, a list of parameters appears:

- **Drum kit mode:** select how to arrange the samples in the sample map.
- **Normal:** put all samples from the starting octave onwards.
- **12 samples per octave:** map the first 12 samples to all octaves, DefleMask-style.
- **Starting octave:** change the octave where the first sample will be at.

following that is a list of viable instrument types. click on one of them to proceed with drum kit creation!

# UI components

---

the user interface consists of several kinds of components, some of which benefit from explanation.

## text fields

---

text fields are able to hold... text.

click on a text field to start editing, and click away to stop editing.

the following keyboard shortcuts work while on a text field:

- Ctrl-X: cut
- Ctrl-C: copy
- Ctrl-V: paste
- Ctrl-A: select all

(replace Ctrl with Command on macOS)

## number input fields

---

these work similar to text fields, but you may only input numbers.

they also usually have + and - buttons which allow you to increase/decrease the value when clicked (and rapidly do so when holding).

additionally, Ctrl-clicking these buttons may increase/decrease the value by a coarse amount.

## sliders

---

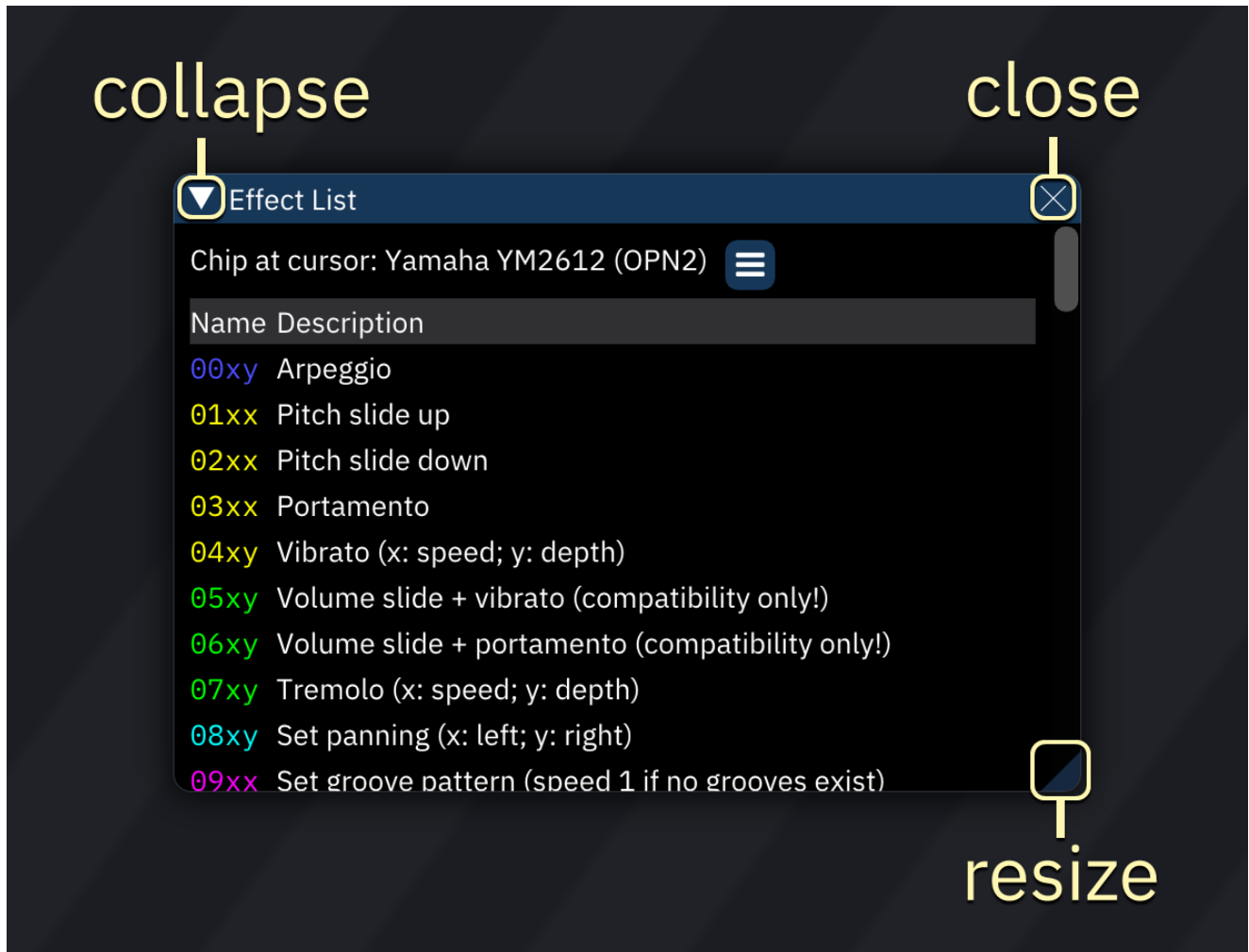
sliders are used for controlling values in a quick manner by being dragged.

using the scroll wheel while holding Ctrl will change the slider's value by small amounts.

right-clicking or Ctrl-clicking or a slider (Command-click on macOS) will turn it into a number input field, allowing you to input precise values.

once you click away it will become a slider again.

# windows



windows may be moved, collapsed, closed or even docked around the workspace.

to move a window, press and hold the left mouse button while on the title bar or any empty space on it.

then drag your mouse, and release it to stop moving.

to resize a window, drag the bottom right corner (marked by a triangular tab) or the borders.

to collapse a window, click on the triangle in the title bar.

clicking again expands the window.

to close a window, click on the X at the top right corner, or select it from the "window" menu.

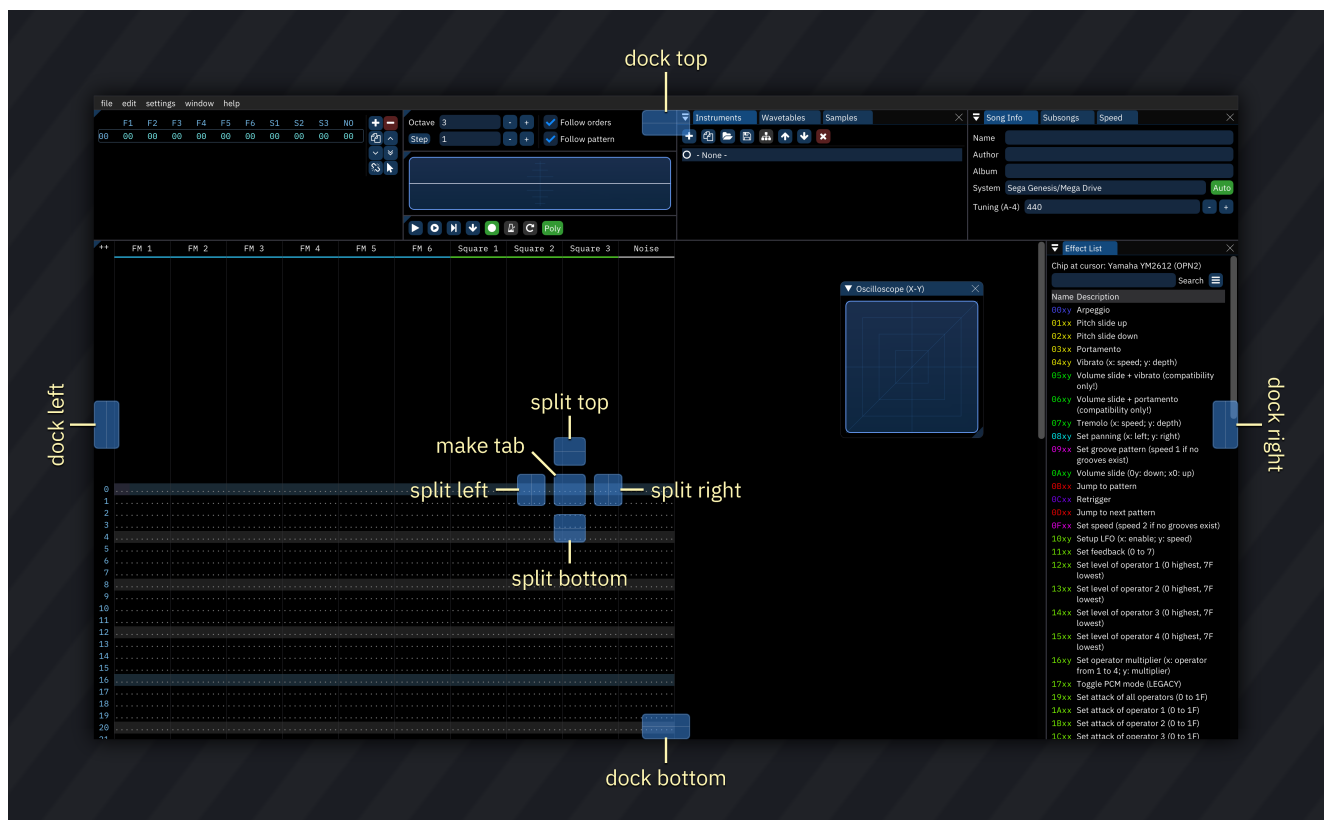
## arrangement and docking

windows may be docked, which comes in handy.

to dock a window, drag it from its title bar to another location in the workspace or to the location of another window.



while dragging, an overlay with some options will appear, allowing you to select where and how to dock that window.  
the options are:



drag your mouse cursor to any of the options to dock the window.

if you drag to the sides, the window will cover that side of the workspace.

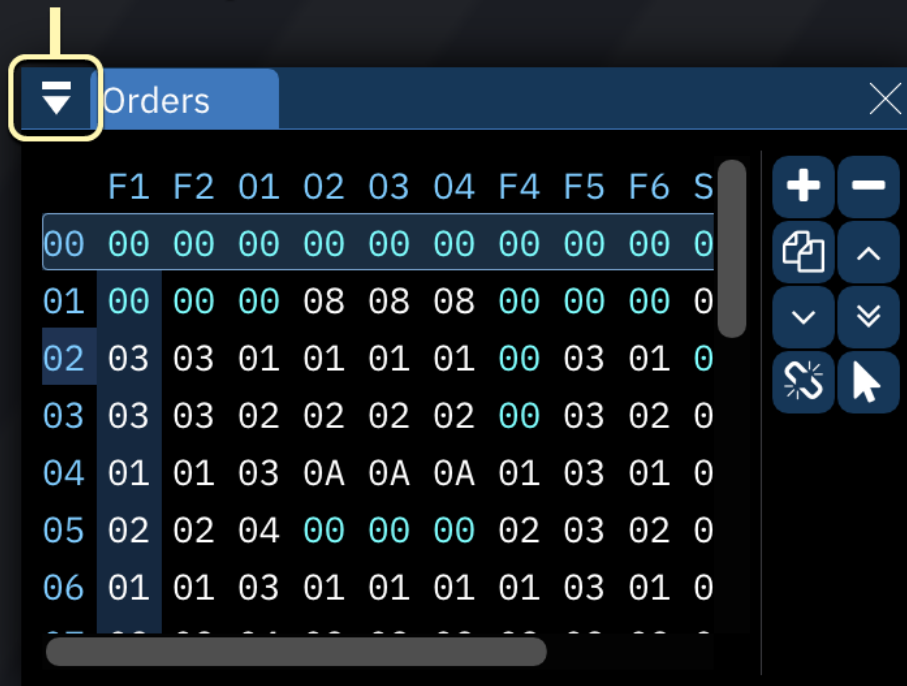
if you drag it to a window or empty space, five docking positions will appear.

if you drag the window to the center of another window, it will appear as another tab.

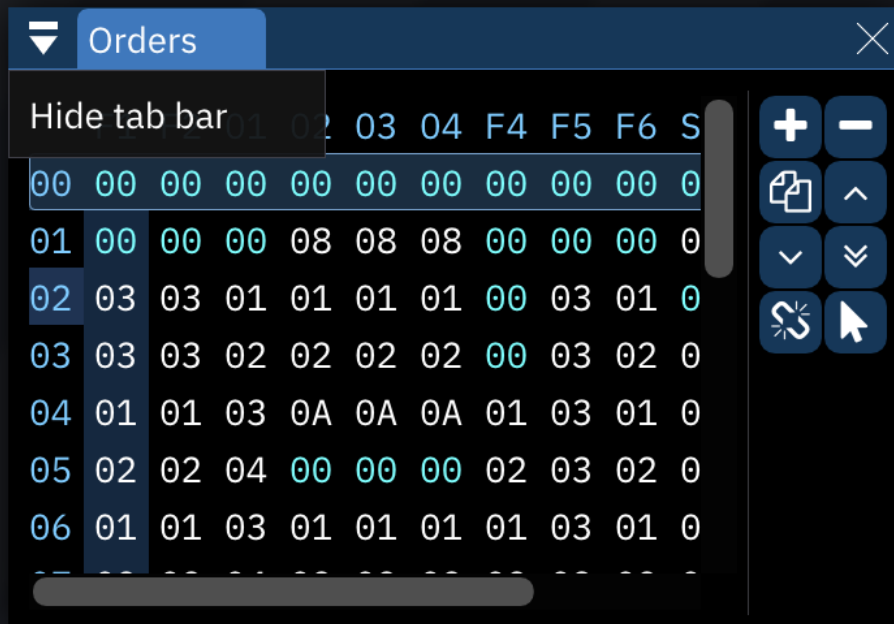
if you drag the window to the center of empty space, the window will cover aforementioned empty space.

otherwise the window will be split in two, with the first half covered by the window you docked and the second half covered by the other window.

# tab list / hide bar



when a window is docked, its title bar turns into a tab bar, and the function provided by the "collapse" triangle at the top left changes.



if this triangle is clicked, a menu will appear with a list of tabs, or a single option if there's only one tab: "Hide tab bar".  
selecting this option will hide the tab bar of that window.

# show tab bar



	F1	F2	01	02	03	04	F4	F5	F6	S1	S
00	00	00	00	00	00	00	00	00	00	01	0
01	00	00	00	08	08	08	00	00	00	02	0
02	03	03	01	01	01	01	00	03	01	01	0
03	03	03	02	02	02	02	00	03	02	02	0
04	01	01	03	0A	0A	0A	01	03	01	05	0
05	02	02	04	00	00	00	02	03	02	00	0
06	01	01	03	01	01	01	01	03	01	00	0
07	02	02	04	02	02	02	02	03	02	00	0
08	01	01	05	03	03	03	03	03	01	00	0

to bring it back, click on the top left corner.

to undock a window, drag its tab away from where it is docked. then it will be floating again.

# effect list window

---



## Effect List



Chip at cursor: PC  
Engine/TurboGrafx-16

Search



Name Description

00xy Arpeggio

01xx Pitch slide up

02xx Pitch slide down

03xx Portamento

04xy Vibrato (x: speed, y: depth)

05xy Volume slide + vibrato (y: depth only!)

06xy Volume slide + portamento (compatibility on)

07xy Tremolo (x: speed, y: depth)

08xy Set panning (x: left, y: right)

09xx Set groove pattern (if no grooves exist)

0Axy Volume slide (0y: down; x0: up)

0Bxx Jump to pattern

0Dxx Jump to next pattern

0Fxx Set speed (speed 2 if no grooves exist)

10xx Set waveform

11xx Toggle noise mode

Effect types to show:



Pitch



Volume



Panning



Song



Time compatibility



Speed



System (Primary)



System (Secondary)



Miscellaneous



All

None

this window provides a list of the effects that are available.

for more details about these effects, see [the effects page](#)<sup>104</sup>.

- **Chip at cursor:** the currently selected chip. the list only shows available effects for this chip.
- menu button: opens a small list of effect categories. toggle each to change whether effects belonging to such categories will be shown in the list.

# export

---

Furnace allows you to export your song in several formats. this section deals with describing the available export options.

## audio

---

this option allows you to export your song in .wav format. I know I know, no .mp3 or .ogg export yet, but you can use a converter.

- **Export type:**
  - **one file:** exports your song to one .wav file.
  - **multiple files (one per chip):** exports the output of each chip to .wav files.
  - **multiple files (one per channel):** exports the output of each channel to .wav files.
    - useful for usage with a channel visualizer such as corrscope.
- **Bit depth:** default is 16-bit integer.
- **Sample rate:** affects the quality of the output file.
  - default is 44100, "CD quality".
  - lower sample rates lose fidelity as upper frequencies disappear.
  - higher sample rates gain frequencies that can't be heard at the cost of file size and rendering time.
- **Channels in file:** default is 2 (stereo). Set to 1 for mono.
- **Loops:** sets the number of times the song will loop.
  - does not have effect if the song ends with FFxx effect.
- **Fade out (seconds):** sets the fade out time when the song is over.
  - does not have effect if the song ends with FFxx effect.

## VGM

---

this option allows exporting to a VGM (Video Game Music) file. these can be played back with VG MPlay (for example).

the following settings exist:

- **format version:** sets the VGM format version to use.
  - versions under 1.70 do not support per-chip volumes, and therefore will ignore the Mixer completely.
  - other versions may not support all chips.
  - use this option if you need to export for a quirky player or parser.
    - for example, RYMCast is picky with format versions. if you're going to use this player, select 1.60.
- **loop:** includes loop information. if disabled, the resulting file won't loop.
- **loop trail:** sets how much of the song is written after it loops.
  - the reason this exists is to work around a VGM format limitation in where post-loop state isn't recorded at all.
  - this may change the song length as it appears on a player.
- **auto-detect:** detect how much to write automatically.
- **add one loop:** add one more loop.



- **custom**: allows you to specify how many ticks to add.
  - 0 is effectively none, disabling loop trail completely.
- this option will not appear if the loop modality isn't set to None as there wouldn't be a need to.
- **add pattern change hints**: this option adds a "hint" when a pattern change occurs. only useful if you're a developer.
- the format of the "hint" data block that gets written is: 67 66 FE 11 11 11 11 01 00 rr pp pp pp ...
  - 11: length, a 32-bit little-endian number
  - 00: order
  - rr: initial row (a 0Dxx effect is able to select a different row)
  - pp: pattern index (one per channel)
- **direct stream mode**: this option allows DualPCM to work. don't use this for other chips.
- may or may not play well with hardware VGM players.
- **chips to export**: select which chips are going to be exported.
- due to VGM format limitations, you can only select up to two of each chip type.
- some chips will not be available, either because VGM doesn't support these yet, or because you selected an old format version.

## ZSM

---

ZSM (ZSound Music) is a format designed for the Commander X16 to allow hardware playback. it may contain data for either YM2151 or VERA chips. Calliope is one of the programs that supports playback of ZSM files.

the following settings are available:

- **Tick Rate (Hz)**: select the tick rate the song will run at.
- I suggest you use the same rate as the song's.
- apparently ZSM doesn't support changing the rate mid-song.
- **loop**: enables loop. if disabled, the song won't loop.
- **optimize size**: removes unnecessary commands to reduce size.

## ROM

---

depending on the system, this option may appear to allow you to export your song to a working ROM image or code that can be built into one. export options are explained in the system's accompanying documentation.

currently, only one system can be exported this way:

- [Atari 2600 \(TlunA\)](#)<sub>310</sub>

## text

---

this option allows you to export your song as a text file.

## command stream

---

this option exports a binary file in Furnace's own command stream format (FCS) which contains a dump of the internal command stream produced when playing the song.

it's not really useful, unless you're a developer and want to use a command stream dump for some reason (e.g. writing a hardware sound driver). see `export - tech.md` in `papers/` for details.

## DMF

---

this option allows you to save your song as a `.dmf` which can be opened in DefleMask.

the following systems are supported when saving as 1.0/legacy (0.12):

- Sega Genesis/Mega Drive (YM2612 + SN76489)
- Sega Genesis/Mega Drive (YM2612 + SN76489, extended channel 3)
- Sega Master System
- Game Boy
- PC Engine
- NES
- Commodore 64
- Arcade (YM2151 + SegaPCM 5-channel compatibility)
- Neo Geo CD (DefleMask 1.0+)

the following systems are also supported when saving as 1.1.3+:

- Sega Master System (with FM expansion)
- NES + Konami VRC7
- Famicom Disk System

only use this option if you really need it. there are many features which DefleMask does not support, such as a variety of effects, FM macros and pitched samples, so these will be lost.

# file formats

---

this is a list of file formats that Furnace supports.

## song/module

---

- Furnace song (.fur)
- import:
- DefleMask module (.dmf)
- FamiTracker module (.ftm/.0cc/.dnm/.eft)
- Amiga tracker module (.mod)
- Scream Tracker 3 module (.s3m)
- FastTracker 2 module (.xm)
- Impulse Tracker module (.it)
- Future Composer module (.fc13/.fc14/.fc/.smod)
- TFM Music Maker module (.tfe)
- export:
- DefleMask module (.dmf)
- VGM (.vgm)
- ZSound Music (.zsm)

## instrument

---

- load/save:
- Furnace instrument (.fui)
- DefleMask preset/patch (.dmp)
- load only:
- TFM Music Maker instrument (.tfi)
- VGM Music Maker instrument (.vgi)
- Scream Tracker 3 instrument (.s3i)
- SoundBlaster instrument (.sbi)
- Wohlstand OPL instrument (.opli)
- Wohlstand OPN instrument (.opni)
- Gens KMod patch dump (.y12)
- BNK file (AdLib) (.bnk)
- FF preset bank (.ff)
- 2612edit GYB preset bank (.gyb)
- VOPM preset bank (.opm)
- Wohlstand WOPL bank (.wopl)
- Wohlstand WOPN bank (.wopn)

## wavetable

---

- load/save:
- Furnace wavetable (.fuw)
- DefleMask wavetable (.dmw)
- raw wavetable data

# sample

---

- load/save
- Wave file (.wav)
- raw sample data
- load only:
- Apple/SGI sample (.aiff)
- Sun/NeXT sample (.au)
- Audio Visual Research sample (.avr)
- Apple Core Audio File sample (.caf)
- HMM Tool Kit sample (.htk)
- Amiga IFF/SVX8/SV16 sample (.iff)
- GNU Octave/Matlab sample (.mat)
- Akai MPC 2k sample (.mpc)
- Ensoniq PARIS sample (.paf)
- Portable Voice Format sample (.pvf)
- RIFF 64 sample (.rf64)
- Sound Designer II sample (.sd2)
- Midi Sample Dump Standard sample (.sds)
- Berkeley/IRCAM/CARL sample (.sf)
- Creative Labs sample (.voc)
- SoundFoundry WAVE 64 sample (.w64)
- NIST Sphere sample (.wav)
- Psion Series 3 sample (.wve)
- FastTracker 2 sample (.xi)
- NES DPCM data (.dmc)
- SNES Bit Rate Reduction (.brr)
- PMD YM2608 ADPCM-B sample bank (.ppc)
- PDR 4-bit AY-3-8910 sample bank (.pps)
- FMP YM2608 ADPCM-B sample bank (.pvi)
- MDX OKI ADPCM sample bank (.pdx)
- FMP 8-bit PCM sample bank (.pzi)
- PMD 8-bit PCM sample bank (.p86)
- PMD OKI ADPCM sample bank (.p)

# keyboard

everything on this list can be configured in the "Keyboard" tab of the Settings dialog.

additionally, everything on this list can be accessed with the "command palette" using the default key combo of `Ctrl-P`.

the keys in the "Global hotkeys" section can be used in any window, although not when a text field is activated.

ACTION	DEFAULT KEYBIND
<b>Global hotkeys</b>	
New	Ctrl-N
Clear song data	—
Open file	Ctrl-O
Restore backup	—
Save file	Ctrl-S
Save as	Ctrl-Shift-S
Export	—
Undo	Ctrl-Z
Redo	Ctrl-Y
Play/Stop (toggle)	Return
Play	—
Stop	—
Play (from beginning)	F5
Play (repeat pattern)	—
Play from cursor	Shift-Return
Step row	Ctrl-Return
Octave up	Keypad *
Octave down	Keypad /
Previous instrument	Shift-Keypad /
Next instrument	Shift-Keypad *
Increase edit step	Ctrl-Keypad *
Decrease edit step	Ctrl-Keypad /
Toggle edit mode	Space
Metronome	Ctrl-M
Toggle repeat pattern	—
Follow orders	—
Follow pattern	—

ACTION	DEFAULT KEYBIND
Toggle full-screen	F11
Request voice from TX81Z	—
Panic	F12
<b>Window activation</b>	
Find/Replace	Ctrl-F
Settings	—
Song Information	—
Subsongs	—
Speed	—
Instrument List	—
Wavetable List	—
Sample List	—
Orders	—
Pattern	—
Mixer	—
Grooves	—
Channels	—
Pattern Manager	—
Chip Manager	—
Compatibility Flags	—
Song Comments	—
Instrument Editor	—
Wavetable Editor	—
Sample Editor	—
Edit Controls	—
Piano	—
Oscilloscope (master)	—
Oscilloscope (per-channel)	—
Oscilloscope (X-Y)	—
Volume Meter	—
Clock	—
Register View	—
Log Viewer	—
Statistics	—
Memory Composition	—
Effect List	—
Debug Menu	Ctrl-Shift-D

ACTION	DEFAULT KEYBIND
Command Stream Player	—
About	—
Collapse/expand current window	—
Close current window	Shift-Escape
Command Palette	Ctrl-P
Recent files (Palette)	—
Instruments (Palette)	—
Samples (Palette)	—
<b>Note input</b>	
<i>see "note input" section after table</i>	
<b>Pattern</b>	
Transpose (+1)	Ctrl-F2
Transpose (-1)	Ctrl-F1
Transpose (+1 octave)	Ctrl-F4
Transpose (-1 octave)	Ctrl-F3
Increase values (+1)	Ctrl-Shift-F2
Increase values (-1)	Ctrl-Shift-F1
Increase values (+16)	Ctrl-Shift-F4
Increase values (-16)	Ctrl-Shift-F3
Select all	Ctrl-A
Cut	Ctrl-X
Copy	Ctrl-C
Paste	Ctrl-V
Paste Mix (foreground)	Ctrl-Shift-V
Paste Mix (background)	—
Paste Flood	—
Paste Overflow	—
Move cursor up	Up
Move cursor down	Down
Move cursor left	Left
Move cursor right	Right
Move cursor up by one (override Edit Step)	Shift-Home
Move cursor down by one (override Edit Step)	Shift-End
Move cursor to previous channel	—
Move cursor to next channel	—
Move cursor to previous channel (overflow)	—

ACTION	DEFAULT KEYBIND
Move cursor to next channel (overflow)	—
Move cursor to beginning of pattern	Home
Move cursor to end of pattern	End
Move cursor up (coarse)	PageUp
Move cursor down (coarse)	PageDown
Expand selection upwards	Shift-Up
Expand selection downwards	Shift-Down
Expand selection to the left	Shift-Left
Expand selection to the right	Shift-Right
Expand selection upwards by one (override Edit Step)	—
Expand selection downwards by one (override Edit Step)	—
Expand selection to beginning of pattern	—
Expand selection to end of pattern	—
Expand selection upwards (coarse)	Shift-PageUp
Expand selection downwards (coarse)	Shift-PageDown
Move selection up by one	Alt-Up
Move selection down by one	Alt-Down
Move selection to previous channel	Alt-Left
Move selection to next channel	Alt-Right
Delete	Delete
Pull delete	Backspace
Insert	Insert
Mute channel at cursor	Alt-F9
Solo channel at cursor	Alt-F10
Unmute all channels	Alt-Shift-F9
Go to next order	—
Go to previous order	—
Collapse channel at cursor	—
Increase effect columns	—
Decrease effect columns	—
Interpolate	—
Fade	—
Invert values	—
Flip selection	—
Collapse rows	—
Expand rows	—
Collapse pattern	—



ACTION	DEFAULT KEYBIND
Expand pattern	—
Collapse song	—
Expand song	—
Set note input latch	—
Clear note input latch	—
Absorb instrument/octave from status at cursor	—
Return cursor to previous jump point	—
Reverse recent cursor undo	—
<b>Instrument list</b>	
Add instrument	Insert
Duplicate instrument	Ctrl-D
Open instrument	—
Open instrument (replace current)	—
Save instrument	—
Save instrument (.dmp)	—
Move instrument up in list	Shift-Up
Move instrument down in list	Shift-Down
Delete instrument	—
Edit instrument	Shift-Return
Instrument cursor up	Up
Instrument cursor down	Down
Instruments: toggle folders/standard view	Ctrl-V
<b>Wavetable list</b>	
Add wavetable	Insert
Duplicate wavetable	Ctrl-D
Open wavetable	—
Open wavetable (replace current)	—
Save wavetable	—
Save wavetable (.dmw)	—
Save wavetable (raw)	—
Move wavetable up in list	Shift-Up
Move wavetable down in list	Shift-Down
Delete wavetable	—
Edit wavetable	Shift-Return
Wavetable cursor up	Up
Wavetable cursor down	Down

ACTION	DEFAULT KEYBIND
Wavetables: toggle folders/standard view	Ctrl-V
<b>Sample list</b>	
Add sample	Insert
Duplicate sample	Ctrl-D
Sample Editor: Create wavetable from selection	Ctrl-W
Open sample	—
Open sample (replace current)	—
Import raw sample data	—
Import raw sample data (replace current)	—
Save sample	—
Save sample (raw)	—
Move sample up in list	Shift-Up
Move sample down in list	Shift-Down
Delete sample	—
Edit sample	Shift-Return
Sample cursor up	Up
Sample cursor down	Down
Sample Preview	—
Stop sample preview	—
Samples: Toggle folders/standard view	Ctrl-V
Samples: Make me a drum kit	—
<b>Orders</b>	
Previous order	Up
Next order	Down
Order cursor left	Left
Order cursor right	Right
Increase order value	—
Decrease order value	—
Switch order edit mode	—
Order: Toggle alter entire row	Ctrl-L
Add order	Insert
Duplicate order	Ctrl-D
Deep clone order	Ctrl-Shift-D
Copy current order to end of song	Ctrl-E
Deep clone current order to end of song	Ctrl-Shift-E
Remove order	Delete

ACTION	DEFAULT KEYBIND
Move order up	Shift-Up
Move order down	Shift-Down
Replay order	—
<b>Sample editor</b>	
Sample editor mode: Select	Shift-I
Sample editor mode: Draw	Shift-D
Sample editor: Cut	Ctrl-X
Sample editor: Copy	Ctrl-C
Sample editor: Paste	Ctrl-V
Sample editor: Paste replace	Ctrl-Shift-V
Sample editor: Paste mix	Ctrl-Alt-V
Sample editor: Select all	Ctrl-A
Sample editor: Resize	Ctrl-R
Sample editor: Resample	Ctrl-E
Sample editor: Amplify	Ctrl-B
Sample editor: Normalize	Ctrl-N
Sample editor: Fade in	Ctrl-I
Sample editor: Fade out	Ctrl-O
Sample editor: Insert silence	Insert
Sample editor: Apply silence	Shift-Delete
Sample editor: Delete	Delete
Sample editor: Trim	Ctrl-Delete
Sample editor: Reverse	Ctrl-T
Sample editor: Invert	Ctrl-Shift-T
Sample editor: Signed/unsigned exchange	Ctrl-U
Sample editor: Apply filter	Ctrl-F
Sample editor: Preview sample	—
Sample editor: Stop sample preview	—
Sample editor: Zoom in	Ctrl-=
Sample editor: Zoom out	Ctrl--
Sample editor: Toggle auto-zoom	Ctrl-0
Sample editor: Create instrument from sample	—
Sample editor: Set loop to selection	Ctrl-L

# menu bar

---

the menu bar allows you to select from five menus: file, edit, settings, window and help.

## file

---

- **new...:** opens the new song dialog to choose a system.
- click a system name to create a new song with it.
- some systems have several variants, which are inside a group.
- **open...:** opens the file picker, allowing you to select a song to open.
- see [file formats](#)<sub>63</sub> for a list of formats Furnace is able to open.
- **open recent:** contains a list of the songs you've opened before.
- **clear history:** erases the file history.
- **save:** saves the current song.
- opens the file picker if this is a new song, or a backup.
- **save as...:** opens the file picker, allowing you to save the song under a different name.
- **export...:** allows you to export your song into other formats, such as audio files, VGM and more. see the [export](#)<sub>60</sub> page for more information.
- **manage chips:** opens the [Chip Manager](#)<sub>360</sub> dialog.
- **restore backup:** restores a previously saved backup.
- Furnace keeps up to 5 backups of a song.
- the backup directory is located in:
  - Windows: %USERPROFILE%\AppData\Roaming\furnace\backups
  - macOS: ~/Library/Application Support/Furnace/backups
  - Linux/other: ~/.config/furnace/backups
- this directory grows in size as you use Furnace. remember to delete old backups periodically to save space.
- **do NOT rely on the backup system as auto-save!** you should save a restored backup because Furnace will not save backups of backups.
- **exit:** closes Furnace.

## edit

---

- **...:** does nothing except prevent accidental clicks on later menu items if the menu is too tall to fit on the program window.
- **undo:** reverts the last action.
- **redo:** repeats what you undid previously.
- **cut:** moves the current selection in the pattern view to clipboard.
- **copy:** copies the current selection in the pattern view to clipboard.
- **paste:** inserts the clipboard's contents in the cursor position.
- you may be able to paste from OpenMPT as well.
- **paste special...:** variants of the paste feature.
- **paste mix:** inserts the clipboard's contents in the cursor position, but does not erase the occupied region.
- **paste mix (background):** does the same thing as paste mix, but doesn't alter content which is already there.
- **paste with ins (foreground):** same thing as paste mix, but changes the instrument.

- **paste with ins (background)**: same thing as paste mix (background), but changes the instrument.
- **paste flood**: inserts the clipboard's contents in the cursor position, and repeats until it hits the end of a pattern.
- **paste overflow**: paste, but it will keep pasting even if it runs over another pattern.
- **delete**: clears the contents in the selection.
- **select all**: changes the selection so it covers a larger area.
  - if the selection is wide, it will select the rows in a column.
  - if the selection is tall, it will select the entire column.
  - if a column is already selected, it will select the entire channel.
  - if a channel is already selected, it will select the entire pattern.
- **operation mask**: toggles which columns will be affected by the listed operations. [more information here](#).<sup>380</sup>
- **input latch**: determines which data are placed along with a note. [more information here](#).<sup>375</sup>
- **note/octave up/down**: transposes notes in the current selection.
- **values up/down**: changes values in the current selection by  $\pm 1$  or  $\pm 16$ .
- **transpose**: transpose notes or change values by a specific amount.
- **interpolate**: fills in gaps in the selection by interpolation between values.
- **change instrument...**: changes the instrument number in a selection.
- **gradient/fade...**: replace the selection with a "gradient" that goes from the beginning of the selection to the end.
  - does not affect the note column.
- **Nibble mode**: when enabled, the fade will be per-nibble (0 to F) rather than per-value (00 to FF).
  - use for effects like 04xy (vibrato).
- **scale...**: scales values in the selection by a specific amount.
  - use to change volume in a selection for example.
- **randomize**: replaces the selection with random values.
  - does not affect the note column.
- **Nibble mode**: when enabled, the randomization will be per-nibble (0 to F) rather than per-value (00 to FF).
- **Set effect::** only appears when the selection includes an effect column. if enabled, an input box will appear. instead of being randomized, all effect types in the selection will be changed to the value entered.
- **invert values**: 00 becomes FF, 01 becomes FE, 02 becomes FD and so on.
- **flip selection**: flips the selection so it is backwards.
- **collapse/expand amount**: allows you to specify how much to collapse/expand in the next two menu items.
- **collapse**: shrinks the selected contents.
- **expand**: expands the selected contents.
- **collapse pattern**: same as collapse, but affects the entire pattern.
- **expand pattern**: same as expand, but affects the entire pattern.
- **collapse song**: same as collapse, but affects the entire song.
  - it also changes speeds and pattern length to compensate.
- **expand song**: same as expand, but affects the entire song.
  - it also changes speeds and pattern length to compensate.
- **find/replace**: shows [the Find/Replace window](#).<sup>368</sup>.
- **clear...**: opens a window that allows you to mass-delete things like songs, unused instruments, and the like.

# settings

---

- **full screen**: expands the Furnace window so it covers your screen.
- **lock layout**: prevents you from dragging/resizing docked windows, or docking more.
- **pattern visualizer**: toggles pattern view particle effects when the song plays.
- **reset layout**: resets the workspace to its defaults.
- **user systems...:** shows the User Systems window. this is detailed in [the User Systems documentation](#)<sup>388</sup>.
- **settings...:** shows the Settings window. these are detailed in [the Settings documentation](#)<sup>82</sup>.

# window

---

all these menu items show or hide their associated windows.

- song
- **song comments**<sup>366</sup>
- **song information**<sup>97</sup>
- **subsongs**<sup>97</sup>
- **channels**<sup>354</sup>
- **chip manager**<sup>360</sup>
- **orders**<sup>76</sup>
- **pattern**<sup>99</sup>
- **pattern manager**<sup>382</sup>
- **mixer**<sup>378</sup>
- **compatibility flags**<sup>367</sup>
- assets
- **instruments**<sup>112</sup>
- **samples**<sup>216</sup>
- **wavetables**<sup>211</sup>
- **instrument editor**<sup>112</sup>
- **sample editor**<sup>216</sup>
- **wavetable editor**<sup>211</sup>
- visualizers
- **oscilloscope**<sup>381</sup>
- **oscilloscope (per-channel)**<sup>356</sup>
- **oscilloscope (X-Y)**<sup>390</sup>
- volume meter
- tempo
- **clock**<sup>362</sup>
- **grooves**<sup>372</sup>
- **speed**<sup>97</sup>
- debug
- **log viewer**<sup>376</sup>
- **register view**<sup>386</sup>
- **statistics**<sup>387</sup>
- **memory composition**<sup>377</sup>
- **effect list**<sup>104</sup>
- **play/edit controls**<sup>78</sup>
- **piano/input pad**<sup>384</sup>

# help

---

- **effect list**: displays the effect list.
- **debug menu**: this menu contains various debug utilities.
- unless you are working with the Furnace codebase, it's not useful.
- **inspector**: this option shows the Dear ImGui Metrics/Debugger window.
- unless you are working with the Furnace codebase, it's not useful.
- **panic**: this resets all chips while the song is playing, effectively silencing everything.
- **about...**: displays the About screen.

at the end of the menu bar, more information may be shown:

- during editing, information about the data under the cursor will be shown here:
- note or note modifier.
- instrument number and name.
- volume in decimal, hex, and percentage.
- effect type and description.
- during playback, these values will be displayed:
- speed/groove @ tick rate (BPM) | order | row | elapsed time
- if any changes or edits have been made but not yet saved, "modified" will appear.

# orders

this window displays the order list. a spreadsheet that contains the order of patterns that will play, from top to bottom.

	F1	F2	01	02	03	04	F4	F5	F6	S1	S2	S3	N0
00	00	00	00	00	00	00	00	00	00	01	00	00	01
01	00	00	00	08	08	08	00	00	00	02	00	00	02
02	03	03	01	01	01	01	00	03	01	01	00	00	01
03	03	03	02	02	02	02	00	03	02	02	00	00	02
04	01	01	03	0A	0A	0A	01	03	01	05	01	00	07
05	02	02	04	00	00	00	02	03	02	00	02	00	08
06	01	01	03	01	01	01	01	03	01	00	01	00	07
07	02	02	04	02	02	02	02	03	02	00	02	00	08
08	04	04	05	03	03	03	03	03	01	00	05	00	07
09	05	05	06	04	04	04	04	03	02	00	00	00	08

along the top are the available channels. their abbreviations can be set in the [channels window](#)<sup>354</sup>. the highlighted channel follows the channel the pattern view cursor is in.

along the left are the order numbers. the highlighted row follows the order the pattern view cursor is in.

each entry in the table is the pattern that will play during that order. these can be changed according to the order edit mode.

hovering over a pattern number will pop up a tooltip showing the name of that pattern, if it has one.

the buttons are:

- **Add new order**: adds a new order.
- **Remove order**: removes the currently selected order.
- **Duplicate order**: adds a new order with patterns matching the selected one directly below it.
- right-click to "deep clone"; this copies all patterns involved to new ones.
- **Move order up**: swaps the selected order with the one above it.
- **Move order down**: swaps the selected order with the one below it.
- **Duplicate order at end of song**: same as "Duplicate order" except the new order is added at the bottom of the list.
- **Order change mode**: selects how much of the order will change with an edit. only applies if "Order edit mode" is set to "Click to change".
- **one**: only current channel's pattern will change.
- **entire row**: all patterns in the order will change.











- **Order edit mode:** selects the method of changing orders.
- **Click to change:** a click will add one to the pattern number. a right-click will subtract one.
- **Select and type (don't scroll):** select a pattern number and type.
- **Select and type (scroll horizontally):** as above, but after entering two digits, the cursor moves to the next channel.
- **Select and type (scroll vertically):** as above, but after entering two digits, the cursor moves to the next order.

# play/edit controls

---

the "Play/Edit Controls" are used to control playback and change parameters of the pattern view.

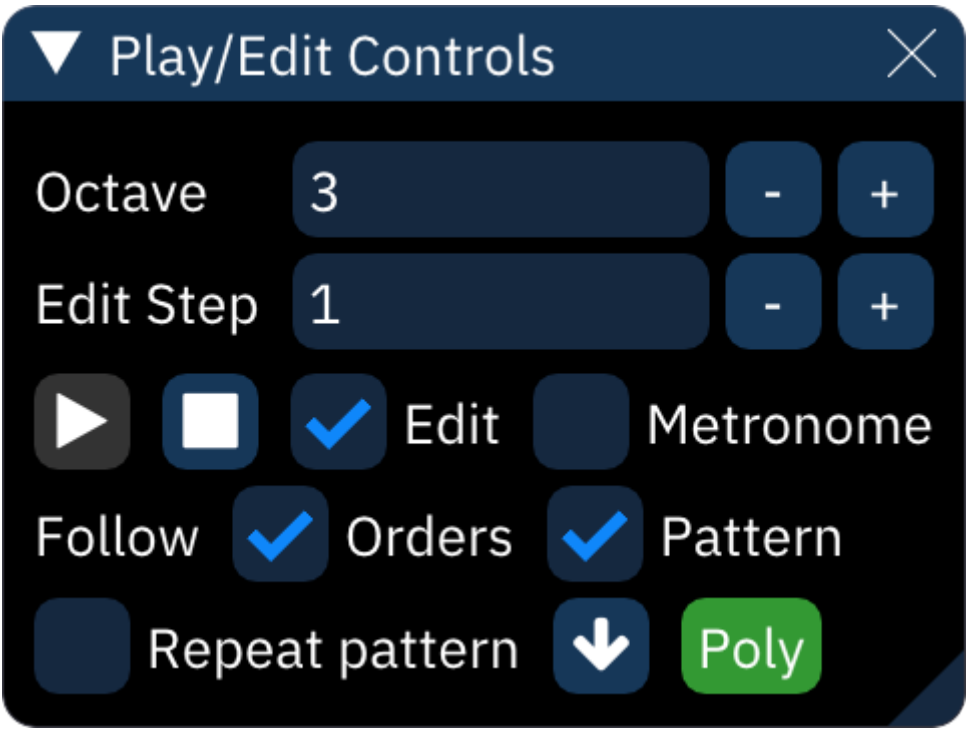
-  **Play:** plays from cursor position.
-  **Stop:** stops all playback.
-  **Play from the beginning of this pattern:** plays from the start of current pattern.
-  **Repeat from the beginning of this pattern:** repeats current pattern from its start.
-  **Step one row:** plays only the row at cursor position and moves down one.
-  **Edit:** toggles edit mode. when turned off, you won't be able to enter notes.
-  **Metronome:** toggles the metronome, which only works during playback.
-  **Repeat pattern:** toggles pattern repeat. during playback while this is on, the current pattern will play over and over instead of following the order list.
- **Poly:** turns on polyphony for previewing notes. toggles to **Mono** for monophony (one note at a time only).
- **Octave:** sets current input octave.
- **Step:** sets edit step. if this is 1, entering a note or effect will move to the next row. if this is a larger number, rows will be skipped. if this is 0, the cursor will stay in place.
- if clicked, Step becomes **Coarse**, which sets the number of rows moved with PgUp, PgDn, and related movement shortcuts. clicking again will revert it to Step.
- **Follow orders:** if on, the selected order in the orders window will follow the song during play back.
- **Follow pattern:** if on, the cursor will follow playback and the song will scroll by as it plays.

## layouts

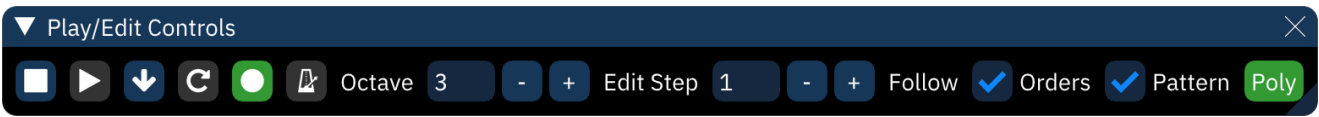
---

the layout can be changed in Settings > Appearance to one of these:

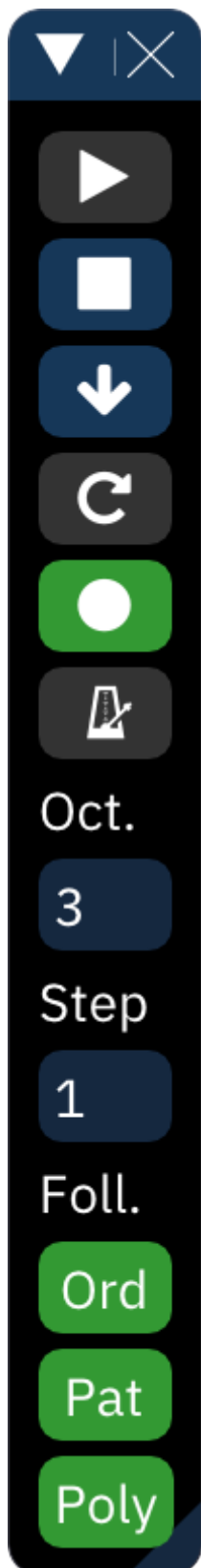
**classic**



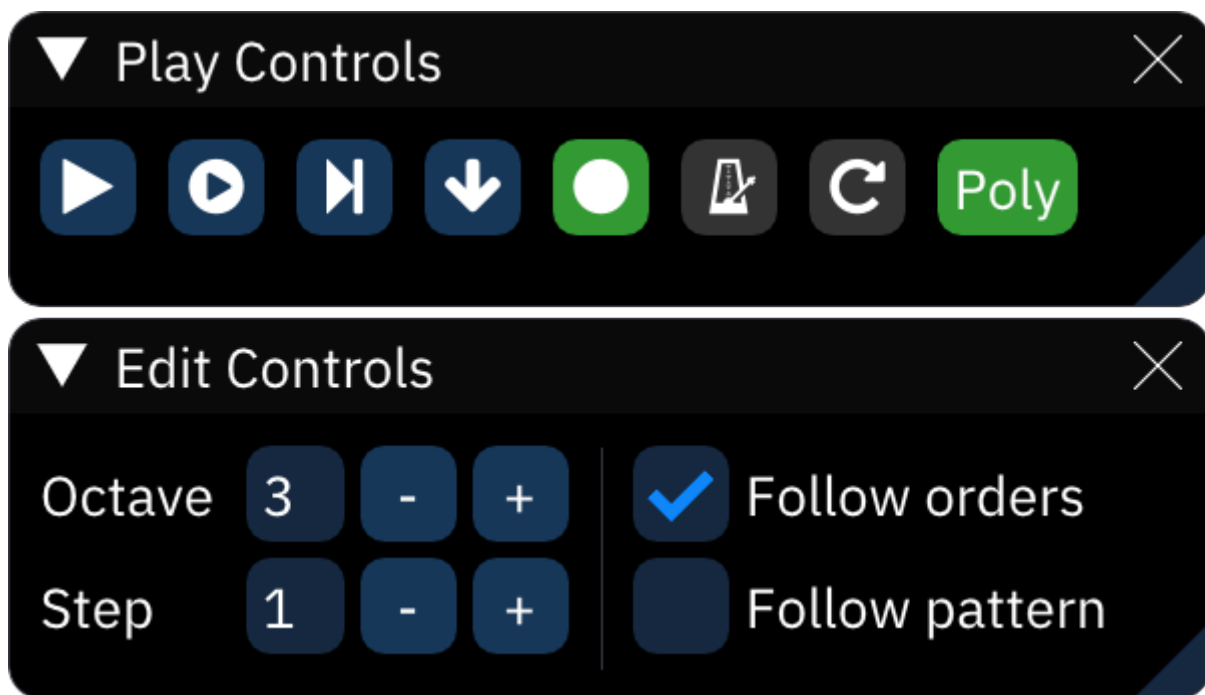
**compact**



## compact (vertical)



## split



# settings

---

the Settings window allows you to change Furnace settings.

settings are saved when clicking the **OK** or **Apply** buttons at the bottom of the window, and when closing the program. several backups are kept in the Furnace settings directory.

## General

---

### Program

- **Language:** select the language used for the interface. some languages are incomplete, and are listed with their approximate completion percentage.
- **Render backend:** changing this may help with performance or compatibility issues. the available render backends are:
  - SDL Renderer: this was the only available render backend prior to the addition of dedicated OpenGL/DirectX backends in 0.6. default on macOS.
    - it is slower than the other backends.
  - DirectX 11: works with the majority of graphics chips/cards and is optimized specifically for Windows.
  - DirectX 9: use if your hardware is incompatible with DirectX 11.
  - OpenGL 3.0: works with the majority of graphics chips/cards (from 2010 onwards). default on Linux.
  - OpenGL 2.0: use if you have a card without OpenGL 3.0 support.
  - OpenGL 1.1: use if your card doesn't even support OpenGL 2.0.
  - Software: this is a last resort backend which renders the interface in software. very slow!
- **Advanced render backend settings:** only applicable with some render backends.
- **Render driver:** this setting only appears when using the SDL Renderer backend. it allows you to select an SDL render driver.
- OpenGL settings: these only appear when using an OpenGL backend, and should only be adjusted if the display is incorrect.
- **VSync:** synchronizes rendering to VBlank and eliminates tearing.
- **Frame rate limit:** allows you to set a frame rate limit (in frames per second).
  - only has effect when VSync is off or not available (e.g. software rendering or force-disabled on driver settings).
- **Display render time:** displays frame rate and frame render time at the right side of the menu bar.
- **Late render clear:** this option is only useful when using old versions of Mesa drivers. it force-waits for VBlank by clearing after present, reducing latency.
- **Power-saving mode:** saves power by lowering the frame rate to 2fps when idle.
  - may cause issues under Mesa drivers!
- **Disable threaded input (restart after changing!):** processes key presses for note preview on a separate thread (on supported platforms), which reduces latency.
- **Enable event delay:** may cause issues with high-polling-rate mice when previewing notes.
- **Per-channel oscilloscope threads:** runs the per-channel oscilloscope in separate threads for a performance boost when there are lots of channels.
- **Oscilloscope rendering engine:** allows you to select between the following rendering engines for oscilloscope views (master and per-channel):
  - **ImGui line plot:** the default engine. uses Dear ImGui line plotting functions for rendering.

- **GLSL**: uses OpenGL shaders for rendering. higher quality, but very GPU-intensive! only available in OpenGL 3.0 backend.

## File

- **Use system file picker**: uses native OS file dialog instead of Furnace's.
- **Number of recent files**: number of files that will be remembered in the *open recent...* menu.
- **Compress when saving**: uses zlib to compress saved songs.
- **Save unused patterns**: stores unused patterns in a saved song.
- **Use new pattern format when saving**: stores patterns in the new, optimized and smaller format. only disable if you need to work with older versions of Furnace.
- **Don't apply compatibility flags when loading .dmf**: does exactly what the option says. your .dmf songs may not play correctly after enabled.
- **Play after opening song**:
  - No
  - Only if already playing
  - Yes
- **Audio export loop/fade out time**:
- **Set to these values on start-up**:
  - **Loops**: number of additional times to play through 0Bxx song loop.
  - **Fade out (seconds)**: length of fade out after final loop.
- **Remember last values**
- **Store instrument name in .fui**: when enabled, saving an instrument will store its name. this may increase file size.
- **Load instrument name from .fui**: when enabled, loading an instrument will use the stored name (if present). otherwise, it will use the file name.
- **Auto-fill file name when saving**: pre-fill the file name field when saving or exporting.
  - when saving a module, the existing file name will be auto-filled.
  - when saving an instrument or sample, its name will be auto-filled.

## New Song

- **Initial system**: the system of chips loaded on starting Furnace.
- **Current system**: sets current chips as default.
- **Randomize**: sets default to a random system.
  - this will not choose a random system at each start.
- **Reset to defaults**: sets default to "Sega Genesis/Mega Drive".
- **Name**: name for the default system. may be set to any text.
- **Configure**: same as in the [chip manager](#)<sup>360</sup> and [mixer](#)<sup>378</sup>.
- **When creating new song**:
- **Display system preset selector**
- **Start with initial system**
- **Default author name**

## Start-up

- **Play intro on start-up**:
- **No**: skips intro entirely.
- **Short**: shows silent title screen briefly.
- **Full (short when loading song)**: shows animated musical intro unless started with a song (command line, double-clicking a .fur file, etc.)

- **Full (always):** always shows animated musical intro.
- **Disable fade-in during start-up**

## Behavior

- **New instruments are blank:** when enabled, adding FM instruments will make them blank (rather than loading the default one).

## Configuration

- **Import:** select an exported .ini config file to overwrite current settings.
- **Export:** select an .ini file to save current settings.
- **Factory Reset:** resets all settings to default and purges settings backups.

## Import

- **Use OPL3 instead of OPL2 for S3M import:** changes which system is used for the import of S3M files that contain FM channels.

# Audio

## Output

- **Backend:** selects a different backend for audio output.
- **SDL:** the default one.
- **JACK:** the JACK Audio Connection Kit (low-latency audio server). only appears on Linux, or MacOS compiled with JACK support.
- **PortAudio:** this may or may not perform better than the SDL backend.
- **Driver:** select a different audio driver if you're having problems with the default one.
- only appears when Backend is SDL.
- **Device:** audio device for playback.
- if using PortAudio backend, devices will be prefixed with the audio API that PortAudio is going to use:
  - **Windows WASAPI:** a modern audio API available on Windows Vista and later, featuring an (optional) Exclusive Mode. be noted that your buffer size setting may be ignored.
  - **Windows WDM-KS:** low-latency, direct to hardware output mechanism. may not work all the time and prevents your audio device from being used for anything else!
  - **Windows DirectSound:** this is the worst choice. best to move on.
  - **MME:** an old audio API. doesn't have Exclusive Mode.
  - **Core Audio:** the only choice in macOS.
  - **ALSA:** low-level audio output on Linux. may prevent other applications from using your audio device.
- **Sample rate:** audio output rate.
- a lower rate decreases quality and isn't really beneficial.
- if using PortAudio backend, be careful about this value.
- **Outputs:** number of audio outputs created, up to 16. default is 2 (stereo).
- **Buffer size:** size of buffer in both samples and milliseconds.
- setting this to a low value may cause stuttering/glitches in playback (known as "underruns" or "xruns").
- setting this to a high value increases latency.
- **Exclusive mode:** enables Exclusive Mode, which may offer latency improvements.



- only available on WASAPI devices in the PortAudio backend!
- **Low-latency mode:** reduces latency by running the engine faster than the tick rate. useful for live playback/jam mode.
- only enable if your buffer size is small (10ms or less).
- **Force mono audio:** use if you're unable to hear stereo audio (e.g. single speaker or hearing loss in one ear).
- **want:** displays requested audio configuration.
- **got:** displays actual audio configuration returned by audio backend.

## Mixing

- **Quality:** selects quality of resampling. low quality reduces CPU load by a small amount.
- **Software clipping:** clips output to nominal range (-1.0 to 1.0) before passing it to the audio device.
- this avoids activating Windows' built-in limiter.
- this option shall be enabled when using PortAudio backend with a DirectSound device.
- **DC offset correction:** apply a filter to remove DC bias, where the output is overall above or below zero. default is on.

## Metronome

- **Volume:** sets volume of metronome.

## Sample preview

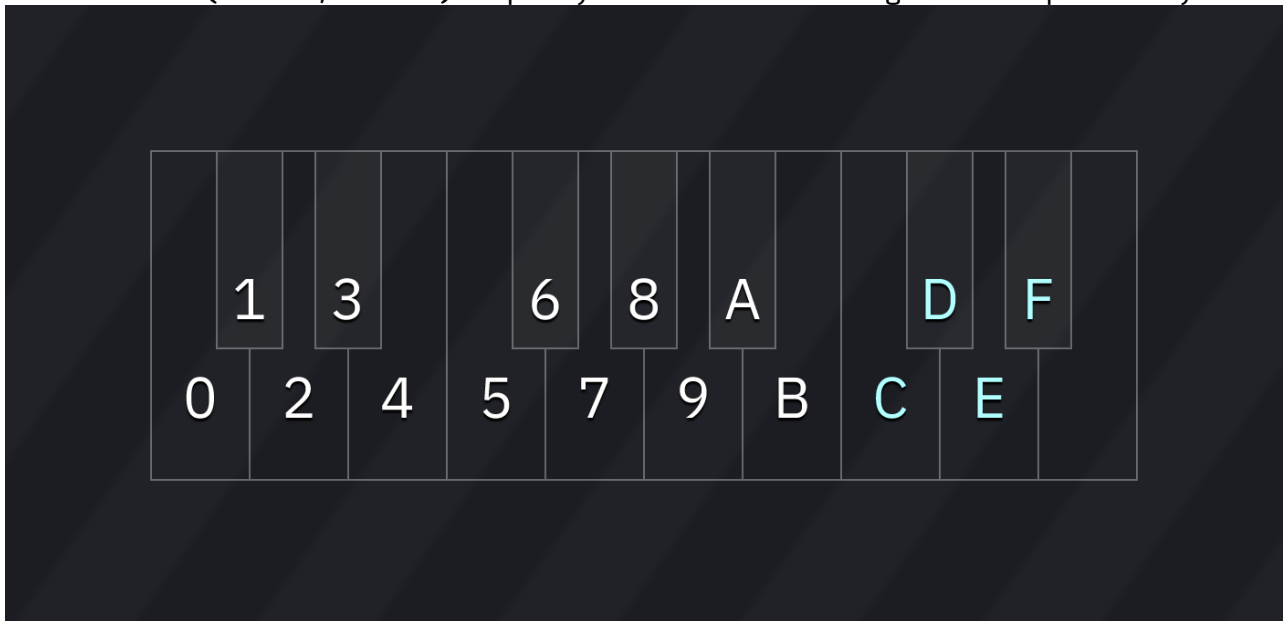
- **Volume:** sets volume of sample preview.

# MIDI

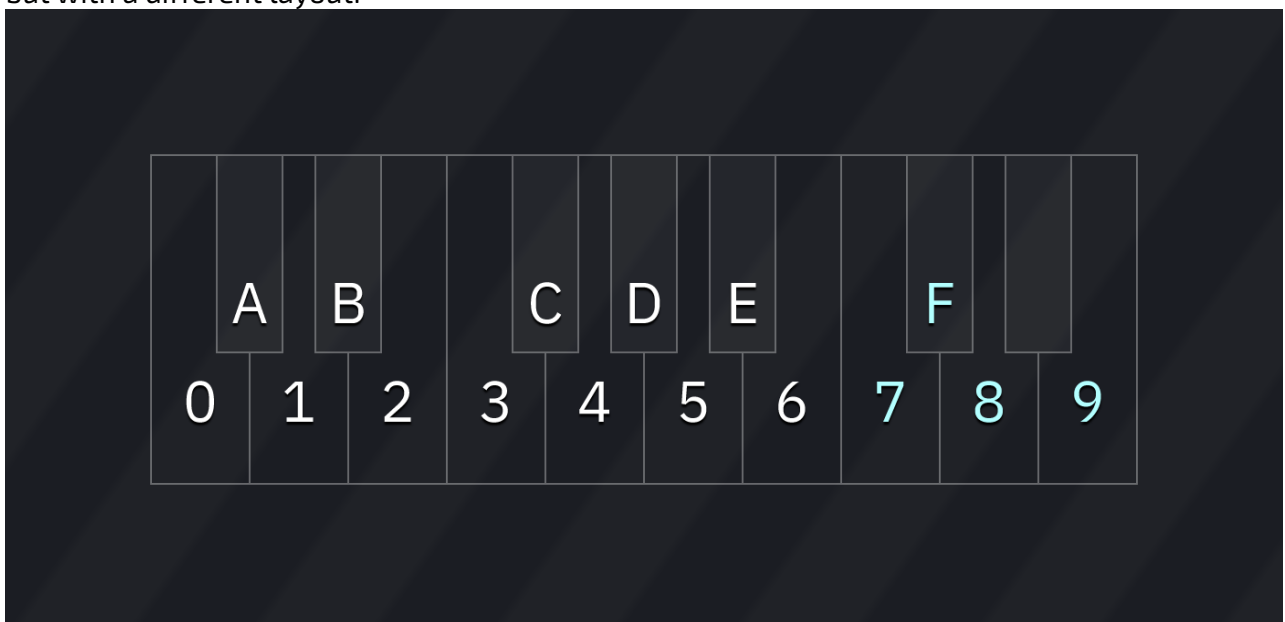
## MIDI input

- **MIDI input:** input device.
- **Rescan MIDI devices:** repopulates list with all currently connected MIDI devices. useful if a device is connected while Furnace is running.
- **Note input:** enables note input. disable if you intend to use this device only for binding actions.
- **Velocity input:** enables velocity input when entering notes in the pattern.
- **Map MIDI channels to direct channels:** when enabled, notes from MIDI channels will be mapped to channels rather than the cursor position.
- **Program change pass-through:** when enabled, program change events are sent to each channel as instrument change commands.
- this option is only available when the previous one is enabled.
- **Map Yamaha FM voice data to instruments:** when enabled, Furnace will listen for any transmitted Yamaha SysEx patches.
- this option is only useful if you have a Yamaha FM synthesizer (e.g. TX81Z).
- selecting a voice or using the "Voice Transmit?" option will send a patch, and Furnace will create a new instrument with its data.
- this may also be triggered by clicking on "Receive from TX81Z" in the instrument editor (OPZ only).
- **Program change is instrument selection:** changes the current instrument when a program change event is received.
- this option is not available when "Program change pass-through" is enabled.

- **Value input style:** changes the way values are entered when the pattern cursor is not in the Note column. the following styles are available:
- **Disabled/custom:** no value input through MIDI.
- **Two octaves (0 is C-4, F is D#5):** maps keys in two octaves to single nibble input. the layout is:



- **Raw (note number is value):** the note number becomes the input value. not useful if you want to input anything above 7F.
- **Two octaves alternate (lower keys are 0-9, upper keys are A-F):** maps keys in two octaves, but with a different layout:



- **Use dual control change (one for each nibble):** maps two control change events to the nibbles of a value.
  - **CC of upper nibble:** select the CC number that will change the upper nibble.
  - **CC of lower nibble:** select the CC number that will change the lower nibble.
- **Use 14-bit control change:** maps two control change events that together form a single 14-bit CC. some MIDI controllers do these.
  - **MSB CC:** select the CC containing the upper portion of the control.
  - **LSB CC:** select the CC containing the lower portion of the control.

- **Use single control change**: maps one control change event. not useful if you want to input odd numbers.
  - **Control**: select the CC number that will change the value.
- **Per-column control change**: when enabled, you can map several control change events to a channel's columns.
- **Volume curve**: adjust the velocity to volume curve.
  - the default is 2.0, which matches General MIDI standard.
- **Actions**: this allows you to bind note input and control change events to actions.
- + button: adds a new action.
- window-with-arrow button: new action with learning! press a button or move a slider/knob/ something on your device.
- each action has the following:
  - **Type**: type of event.
  - **Channel**: channel of event.
  - **Note/Control**: the note/control change number.
  - **Velocity/Value**: the velocity or control value
  - **Action**: the GUI action to perform.
  - **Learn**: after clicking on this button, do something in your MIDI device and Furnace will map that to this action.
  - **Remove**: remove this action.

## MIDI output

- **MIDI output**: output device.
- **Output mode**:
  - **Off (use for TX81Z)**: don't output anything. use if you plan to use Furnace as sync master, or the "Receive from TX81Z" button in the OPZ instrument editor.
  - **Melodic**: output MIDI events.
  - **Send Program Change**: output program change events when instrument change commands occur.
  - **Send MIDI clock**: output MIDI beat clock.
  - **Send MIDI timecode**: output MIDI timecode.
  - **Timecode frame rate**: sets the timing standard used for MIDI timecode.
    - **Closest to Tick Rate**: automatically sets the rate based on the song's Tick Rate.
    - **Film (24fps)**: output at 24 codes per second.
    - **PAL (25fps)**: output at 25 codes per second.
    - **NTSC drop (29.97fps)**: output at ~29.97 codes per second, skipping frames 0 and 1 of each minute that doesn't divide by 10.
    - **NTSC non-drop (30fps)**: output at 30 codes per second.

## Emulation

---

### Cores

- **Playback Core(s)**: core(s) to use for playback.
- **Render Core(s)**: core(s) to use when exporting audio.

all of these are covered in the [guide to choosing emulation cores](#)<sup>392</sup>.

## Quality

some chips have output quality settings. these may be used to increase quality or lower CPU usage.

the available quality settings are:

- Lower: fastest but worst quality.
- Low
- Medium
- High: the default quality.
- Ultra
- Ultimate: highest available quality. may be very CPU heavy!

## Other

- **PC Speaker strategy:** this is covered in the [PC speaker page](#)<sup>278</sup>.

## Keyboard

---

### Keyboard

- **Import:** imports keyboard layout in .cfgk format.
- **Export:** exports keyboard layout in .cfgk format.
- **Reset defaults:** resets all keybinds to default values.

a list of keybinds is displayed.

- click on a keybind. then enter a key or key combination to change it.
- right-click to clear the keybind.
- the full list is in the [keyboard](#)<sup>65</sup> page.

### note input

the settings for note input keybinds operate differently. each entry in the list of keybinds is made of the following:

- **Key:** key assignment.
- **Type:** type of note input. left-click cycles through "Note", "Note off", "Note release", and "Macro release".
- **note:** the list is sorted by type. on changing a key's type, it will instantly move to its new sorting position!
- **Value:** number of semitones above C at the current octave. only appears for note type binds.
- **Remove:** removes the keybind from the list.

below all the binds, select a key from the dropdown list to add it. it will appear at or near the top of the list as a note with value 0.

## Interface

---

### Layout

- **Workspace layout**

- **Import:** reads a .ini layout file.
- **Export:** writes current layout to a .ini file.
- **Reset:** resets layout to default.
- **Allow docking editors:** when enabled, you'll be able to dock instrument/wave/sample editors.
- **Remember window position:** remembers the window's last position on start-up.
- **Only allow window movement when clicking on title bar**
- **Center pop-up windows**
- **Play/edit controls layout:**
- **Classic**
- **Compact**
- **Compact (vertical)**
- **Split**
- **Position of buttons in Orders:**
- **Top**
- **Left**
- **Right**

## Mouse

- **Double-click time (seconds):** maximum time between mouse clicks to recognize them as a double-click.
- **Don't raise pattern editor on click**
- **Focus pattern editor when selecting instrument**
- **Draggable instruments/samples/waves:** allow dragging and dropping assets within their lists to reorder them.
- **Note preview behavior:** allows you to disable note preview when entering notes in the pattern.
- **Never:** don't preview notes at all.
- **When cursor is in Note column:** only when the cursor is in the Note column
- **When cursor is in Note column or not in edit mode:** erm... yeah.
- **Always:** always preview notes.
- **Allow dragging selection:**
- **No:** don't allow drag-and-drop.
- **Yes:** allow drag-and-drop.
- **Yes (while holding Ctrl only):** allow drag-and-drop but only when holding Control (Command on macOS).
- **Toggle channel solo on:** selects which interactions with a channel header will toggle solo for that channel.
- **Right-click or double click**
- **Right-click**
- **Double-click**
- **Modifier for alternate wheel-scrolling (vertical/zoom/slider-input):** selects which key to hold for alternate scrolling of interface elements that support it.
- **Ctrl or Meta/Cmd**
- **Ctrl**
- **Meta/Cmd**
- **Alt**
- **Double click selects entire column:** when enabled, double clicking on a cell of the pattern will select the entire column.

## Cursor behavior

- **Insert pushes entire channel row:** when enabled, pressing Insert will push the entire channel rather than the column at the cursor position.
- **Pull delete affects entire channel row:** when enabled, pull deleting (Backspace by default) will pull the entire channel rather than the column at the cursor position.
- **Push value when overwriting instead of clearing it:** in the order list and pattern editors, typing into an already-filled value will shift digits instead of starting fresh. for example:
  - if off: moving the cursor onto the value A5 and typing a "B" results in 0B.
  - if on: moving the cursor onto the value A5 and typing a "B" results in 5B.
- **Keyboard note/value input repeat (hold key to input continuously)**
- **Effect input behavior:**
  - **Move down:** after entering an effect (or effect value), the cursor moves down.
  - **Move to effect value (otherwise move down):** after entering an effect, the cursor moves to its value. if entering a value, the cursor moves down.
  - **Move to effect value/next effect and wrap around:** after entering an effect or effect value, the cursor moves right. if it was on the last column, it jumps back to the first effect.
  - **Delete effect value when deleting effect:** if enabled, deleting effect will also delete its value.
  - **Change current instrument when changing instrument column (absorb):** if enabled, typing on the instrument column will also select the instrument you've typed.
  - **Remove instrument value when inserting note off/release:** if enabled, inserting a note off or release on a row that has instrument value will remove the instrument value.
  - **Remove volume value when inserting note off/release:** same as above, but for volume.

## Cursor movement

- **Wrap horizontally:** selects what to do when the cursor hits horizontal boundaries.
  - **No:** don't wrap the cursor.
  - **Yes:** wrap the cursor.
  - **Yes, and move to next/prev row:** wrap the cursor and move it to the other row.
- **Wrap vertically:** selects what to do when the cursor hits vertical boundaries.
  - **No:** don't wrap the cursor.
  - **Yes:** wrap the cursor.
  - **Yes, and move to next/prev pattern:** wrap the cursor and go to the next/previous order.
  - **Yes, and move to next/prev pattern (wrap around):** same as the previous option, but also wraps around the song.
- **Cursor movement keys behavior:** allows you to select how much will the cursor move by when pressing cursor movement keys.
  - **Move by one:** guess.
  - **Move by Edit Step:** guess.
  - **Move cursor by edit step on delete:** when deleting, moves the cursor by Edit Step.
  - **Move cursor by edit step on insert (push):** when inserting, moves the cursor by Edit Step.
  - **Move cursor up on backspace-delete:** when pull deleting (Backspace by default), moves cursor up.
  - **Move cursor to end of clipboard content when pasting:** allows you to choose what happens after pasting.
    - if on, the cursor will move to the end of the clipboard content.
    - if off, the cursor won't move.

## Scrolling

- **Change order when scrolling outside of pattern bounds:**
- **No:** the pattern edit cursor will stay locked within the current order.
- **Yes:** moving the cursor past the edge of the previous or next order will move to that order, but not past the start or end of a song.
- **Yes, and wrap around song:** as above, but will wrap from song end to start.
- **Cursor follows current order when moving it**
- applies when playback is stopped.
- **Don't scroll when moving cursor**
- **Move cursor with scroll wheel**
- **No**
- **Yes**
- **Inverted**

## Assets

- **Display instrument type menu when adding instrument**
- if turned off, the menu can still be opened by right-clicking the add button.
- **Select asset after opening one**

## Appearance

---

### Scaling

- **Automatic UI scaling factor:** automatically matches the OS's UI scaling.
- **UI scaling factor:** only appears if "Automatic UI scaling factor" is off.
- **Icon size**

### Text

- **Font renderer:** this allows you to select which font renderer library to use. there are two choices:
- **stb\_truetype:** the original font renderer used in Furnace 0.6 and before.
- **FreeType:** this font renderer has support for more font formats and provides more settings. introduced in Furnace 0.6.1.
- **Main font:** font for the user interface.
- **Header font:** font for section headers.
- **Pattern font** font for the pattern view, the order list, and related.
- if "Custom...", a file path selector will appear.
- **Size:** font size.

#### FreeType-specific settings

- **Anti-aliased fonts:** when enabled, fonts will be rendered smooth.
- **Support bitmap fonts:** this option allows you to enable the loading of bitmap fonts.
- be noted that this may force non-bitmap fonts to undesired sizes!
- **Hinting:** this option allows you to define how crisp fonts are rendered.
- **Off:** disable font hinting. at small sizes, fonts may be blurry.
- **Slight:** enable slight font hinting.
- **Normal:** enable font hinting.

- **Full:** enable harsh font hinting. fonts may look ugly/distorted to some people.
- **Auto-hinter:** some fonts contain hinting data, but some don't. this allows you to select what happens.
- **Disable:** only rely upon font hinting data.
- **Enable:** prefer font hinting data if present.
- **Force:** ignore font hinting data.

#### non-specific settings

- **Oversample:** renders the font internally at higher resolution for visual quality.
- higher settings use more video memory.
- for pixel or bitmap fonts, set this to **1x**.
- **Load fallback font:** load an extra font that contains nearly all characters that can be used, in case the selected fonts lack them. uses much video memory
- **Load fallback font (pattern):** as "Load fallback font" above but for the pattern font.
- **Display Japanese characters, Display Chinese (Simplified) characters, Display Chinese (Traditional) characters** and **Display Korean characters:** only toggle these options if you have enough graphics memory.
- these are a temporary solution until dynamic font atlas is implemented in Dear ImGui.

## Program

- **Title bar:**
- **Furnace**
- **Song Name - Furnace**
- **file\_name.fur - Furnace**
- **/path/to/file.fur - Furnace**
- **Display system name on title bar**
- **Display chip names instead of "multi-system" in title bar**
- **Status bar:**
- **Cursor details**
- **File path**
- **Cursor details or file path**
- **Nothing**
- **Display playback status when playing:** display playback time and current location in the menu bar.
- **Export options layout:**
- **Sub-menus in File menu:** export options appear in the File menu as sub-menus.
- **Modal window with tabs:** a single "export..." option that opens a dialog with export options. this is the default.
- **Modal windows with options in File menu:** like Sub-menus in File menu, but instead of being sub-menus, selecting one opens a dialog with export settings.
- **Capitalize menu bar**
- **Display add/configure/change/remove chip menus in File menu:** if enabled, the "manage chips" item in the file menu is split into the four listed items for quick access.

## Orders

- **Highlight channel at cursor in Orders**
- **Orders row number format:**
- **Decimal**



- Hexadecimal

## Pattern

- **Center pattern view:** centers pattern horizontally in view.
- **Overflow pattern highlights**
- **Display previous/next pattern**
- **Pattern row number format:**
- **Decimal**
- **Hexadecimal**
- **Pattern view labels:**
- **Note off (3-char):** default is OFF
- **Note release (3-char):** default is ==.
- **Macro release (3-char):** default is REL.
- **Empty field (3-char):** default is . . .
- **Empty field (2-char):** default is . .
- **Pattern view spacing after:** number of pixels of space between columns.
- **Note**
- **Instrument**
- **Volume**
- **Effect**
- **Effect value**
- **Single-digit effects for 00-0F**
- **Use flats instead of sharps**
- **Use German notation:** display B notes as H, and A# notes as B.

## Channel

- **Channel style:** sets the appearance of channel headers in pattern view.
- **Classic**
- **Line**
- **Round**
- **Split button**
- **Square border**
- **Round border**
- **Channel volume bar:**
- **None**
- **Simple**
- **Stereo**
- **Real**
- **Real (stereo)**
- **Channel feedback style:**
- **Off**
- **Note**
- **Volume**
- **Active**
- **Channel font:**
- **Regular**
- **Monospace**
- **Center channel name**
- **Channel colors:**

- **Single**
- **Channel type**
- **Instrument type**
- **Channel name colors:**
- **Single**
- **Channel type**
- **Instrument type**

## Assets

- **Unified instrument/wavetable/sample list:** combines all three types of assets into one list.
- the buttons act as appropriate to the currently selected asset or header.
- **Horizontal instrument/wavetable list:** when there are more instruments/wavetables than there is room to display them...
- if on, scroll horizontally through multiple columns.
- if off, scroll vertically in one long column.
- only appears if "Unified instrument/wavetable/sample list" is off.
- **Instrument list icon style:**
- **None**
- **Graphical icons**
- **Letter icons**
- **Colorize instrument editor using instrument type**

## Macro Editor

- **Macro editor layout:**
- **Unified**
- **Grid**
- **Single (with list)**
- **Use classic macro editor vertical slider**
- **Macro step size/horizontal zoom:**
- **Manual**
- **Automatic per macro**
- **Automatic (use longest macro)**

## Wave Editor

- **Use compact wave editor**

## FM Editor

- **FM parameter names:**
- **Friendly**
- **Technical**
- **Technical (alternate)**
- **Use standard OPL waveform names**
- **FM parameter editor layout:**
- **Modern**
- **Compact (2x2, classic)**
- **Compact (1x4)**
- **Compact (4x1)**

- **Alternate (2x2)**
- **Alternate (1x4)**
- **Alternate (4x1)**
- **Position of Sustain in FM editor:**
- **Between Decay and Sustain Rate**
- **After Release Rate**
- **After Release Rate, after spacing**
- **After TL**
- **Use separate colors for carriers/modulators in FM editor**
- **Unsigned FM detune values:** uses the internal representation of detune values, such that detune amounts of -1, -2, and -3 are shown as 5, 6, and 7.

## Memory Composition

- **Chip memory usage unit:** unit for displaying memory usage in the Memory Composition window.
- **Bytes**
- **Kilobytes**

## Oscilloscope

- **Rounded corners**
- **Border**
- **Mono:** displays a single monaural waveform of all sound mixed together.
- if turned off, waves will be drawn on top of each other for each output channel.
- all colors are configurable via *Settings > Color > Color scheme > Oscilloscope > Wave (non-mono)*.
- **Anti-aliased:** smoothes the lines of the waveform.
- slight performance cost and slightly buggy.
- **Fill entire window:** removes the gap between the waveform and the edge of the window.
- **Waveform goes out of bounds:** allows the waveform to draw past the top and bottom of the oscilloscope.
- **Line size:** line thickness.
- **Per-channel oscilloscope threads:** number of CPU threads allocated to handle individual oscilloscopes. a reasonable setting is the total number of available cores minus two.
- **Oscilloscope rendering engine:** chooses which line-drawing method to use.
- **ImGui line plot:** use the UI's native rendering method. default.
- **GLSL (if available):** render using shaders that run on the graphics card. only available when using the OpenGL 3.0 render backend.

## Song Comments

- **Wrap text:** visually breaks long lines at the width of the text box. does not affect the text itself.

## Windows

- **Rounded window corners**
- **Rounded buttons**
- **Rounded menu corners**
- **Rounded tabs**
- **Rounded scrollbars**

- **Borders around widgets:** draws borders on buttons, checkboxes, text widgets, and the like.

## Color

---

### Color scheme

- **Import**
- **Export**
- **Reset defaults**
- **Guru mode:** exposes all color options (instead of accent colors).
- **Interface**
- **Frame shading:** applies a gradient effect to buttons and input boxes.
- **Color scheme type:**
  - **Dark**
  - **Light**
- **Accent colors:** select main interface colors.
  - **Primary**
  - **Secondary**
- several more categories...

## Backup

---

### Configuration

- **Enable backup system:** turn on automatic backups of the current open file.
- **Interval (in seconds):** time between automatic backups.
- **Backups per file:** maximum number of backups to store for each file. oldest backups are deleted first.

### Backup Management

- **Purge before:**
- **Go:** purge all backups from before the selected date.
- total space used by all backups:
- **Refresh:** recalculate space.
- **Delete All:** purge all backups.

# song info

---

- **Name:** the track's title.
- **Author:** the author(s) of this track.
- **Album:** the associated album name (or the name of the game the song is from).
- **System:** the name of the game console or computer the track is designed for. this is automatically set when creating a new tune, but can be changed to anything. the **Auto** button will provide a guess based on the chips in use.

all of this metadata will be included in a VGM export. this isn't the case for an audio export, however.

- **Tuning (A-4):** set tuning based on the note A-4, which should be 440 in most cases. opening an Amiga MOD will set it to 436 for hardware compatibility.

## subsongs

---

this window allows one to create **subsongs** - multiple individual songs within a single file. each song has its own order list and patterns, but all songs within a file share the same chips, samples, and so forth.

- the drop-down box selects the current subsong.
- the + button adds a new subsong.
- the - button permanently deletes the current subsong (unless it's the only one).
- **Name:** title of the current subsong.
- the box at the bottom can store any arbitrary text, like a separate "Comments" box for the current subsong.

## speed

---

there are multiple ways to set the tempo of a song.

**Base Tempo:** tempo in beats per minute (BPM). this is affected by the Highlight settings below.

- clicking the Base Tempo button switches to the more technical Tick Rate.

**Tick Rate:** the frequency of ticks per second, thus the rate at which notes and effects are processed.

- all values are allowed for all chips, though most chips have hardware limitations that mean they should stay at either 60 (approximately NTSC) or 50 (exactly PAL).
- clicking the Tick Rate button switches to the more traditional Base Tempo BPM setting.

**Speed:** the number of ticks per row.

- clicking the "Speed" button changes to more complex modes covered in the [grooves<sub>372</sub>](#) page.

**Virtual Tempo:** simulates any arbitrary tempo without altering the tick rate. it does this by adding or skipping ticks to approximate the tempo. the two numbers represent a ratio applied to the actual tick rate. example:

- set tick rate to 150 BPM (60 Hz) and speed to 6.
- set the first virtual tempo number (numerator) to 200.
- set the second virtual tempo number (denominator) to 150.
- the track will play at 200 BPM.
- the ratio doesn't have to match BPM numbers. set the numerator to 4 and the denominator to 5, and the virtual BPM becomes  $150 \div 4/5 = 120$ .
- another way to accomplish this with more control over the results is to use grooves. see the page on [grooves](#)<sup>372</sup> for details.

**Divider:** changes the effective tick rate. a tick rate of 60Hz and a divisor of 6 will result in ticks lasting a tenth of a second each!

- to the right, the effective BPM is listed, taking all settings into account.

**Highlight:** sets the pattern row highlights:

- the first value represents the number of rows per beat.
- the second value represents the number of rows per measure.
- these don't have to line up with the music's actual beats and measures. set them as preferred for tracking.
- note: these values are used for the metronome and calculating BPM.

**Pattern Length:** the length of each pattern in rows. this affects all patterns in the song, and every pattern must be the same length. (Individual patterns can be cut short by 0Bxx, 0Dxx, and FFxx commands.)

**Song Length:** how many orders are in the order list. decreasing it will hide the orders at the bottom. increasing it will restore those orders; increasing it further will add new orders of all 00 patterns.

# pattern

the pattern view allows you to edit the song's patterns.

channel

previous order

cursor

row number

selection

	FM 5	FM 6	Square 1
60			
61			
62			
63			
0	OFF	C-3037B0256	C#3040B00000400
1	F#30A66E578141D0801		0A
2		G-8056D0200	09
3	C#40A66E578141D0801	C-3036B0256	08
4		G-8056D0200	07
5		OFF	08
6		E-506770299	090464
7		OFF	0A
8		E-5067B0299	0B
9	161C		0C
10	161B	G-8056D0200	0D
11	0463161A	E-5066B0299	0C
12	1619	C-303790256	0B
13	1618	OFF	
14	1617	C-303710256	0A
15	1616		
16	1615	C-3037B0256	09

note volume effect type effect value

F#30A66E578141D0801

instrument effect effect effect

a pattern consists of columns ("channels") and numbered rows.  
each column has several subcolumns in this order:

1. note

2. instrument
3. volume
4. effects, split into effect type and effect value

all columns are represented in hexadecimal, except for the note column.

row highlights show beats and measures, and are configured in the [the Speed window](#)<sup>97</sup>.

## cursor and selection

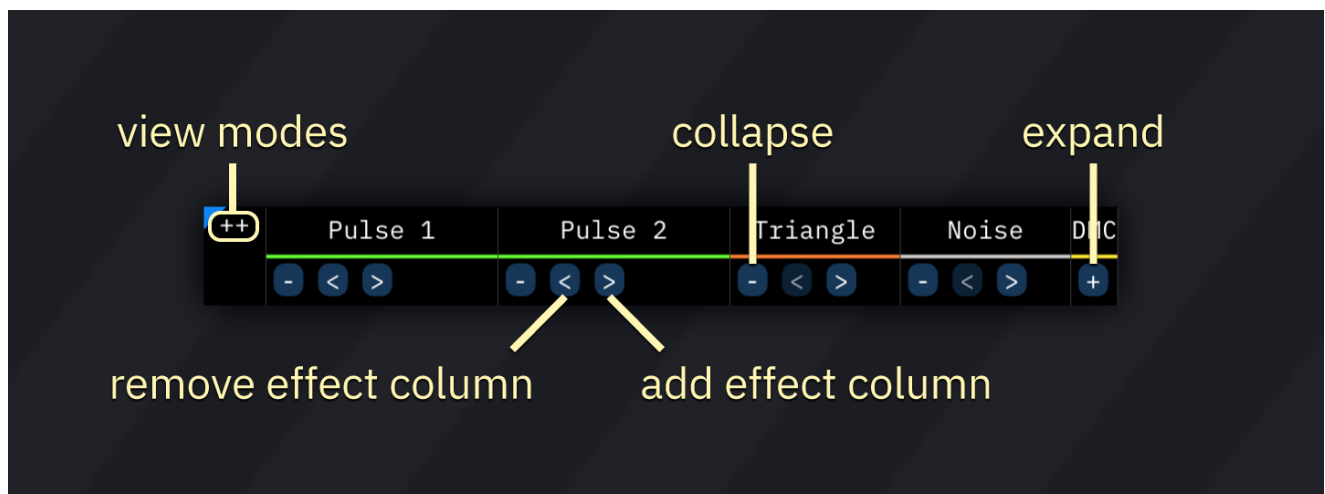
you may change the cursor position by clicking anywhere on the pattern.

to select an area, press and hold the left mouse button. then drag the mouse and release the button to finish selection.

right-clicking within the pattern view brings up a pop-up menu with most options from the [edit menu](#)<sup>72</sup>.

## channel bar

using the channel bar, you may adjust several aspects of the channel display.



clicking on a channel name mutes that channel.

double-clicking or right-clicking it enables solo mode, in which only that channel will be audible.

clicking the ++ at the top left corner of the pattern view pops up a small menu to set view modes:

- **Effect columns/collapse:** displays extra options for collapsing channels and adding/removing effect columns:
  - -: collapse visible columns. changes to + when columns are hidden; click to expand them.
  - <: disables the last effect column and hides it. effects are not deleted...
  - >: adds an effect column. if one previously existed, its contents will be preserved.
- **Pattern names:** displays pattern names (per channel). pattern names are also visible when hovering over a pattern in the order list.
- **Channel group hints:** display indicators when channels are paired in some way (e.g. OPL3 4-op mode).



- **Visualizer:** during playback, show visual effects in the pattern view.
- also can be toggled by right-clicking on the ++ button.
- **Channel status:** displays icons that indicate activity in the channel. see the "channel status" section below.

to rename and/or hide channels, open [the Channels window](#)<sup>354</sup> via the window menu.

## channel status

- note status:
  -  note off
  -  note on
  -  note on but macro released (REL)
  -  note released (===)
- pitch alteration:
  -  nothing
  -  pitch slide up
  -  pitch slide down
  -  portamento
  -  arpeggio
- volume alteration:
  -  nothing
  -  volume slide up
  -  volume slide down
  -  tremolo
- other icons may be present depending on the used chips.

# input

## note input



- pressing any of the respective keys will insert a note at the cursor's location, then advance to the next row (or otherwise according to the Edit Step.)
- **note off** (OFF) turns off the last played note in that channel (key off for FM/hardware envelope; note cut otherwise).
- **note release** (===) triggers macro release (and in FM/hardware envelope channels it also triggers key off).
- **macro release** (REL) does the same as above, but does not trigger key off in FM/hardware envelope channels.
- **toggle edit** enables and disables editing. when editing is enabled, the cursor's row will be shaded red.

## instrument/volume input

type any hexadecimal number (0-9 and A-F). the cursor will move by the Edit Step when a suitable value is entered.

## effect input

works like the instrument/volume input.

each effect column has two subcolumns: effect and effect value.  
if the effect value is not present, it is treated as 00.

most effects run until canceled using an effect of the same type with effect value 00, with some exceptions.

here's a [list of effect types](#)<sup>104</sup>.

# keyboard shortcuts

---

these are the default key functions. all keys are configurable in the Keyboard tab of the Settings window.

KEY	ACTION
Up/Down	move cursor up/down by one row or the Edit Step (configurable)
Left/Right	move cursor left/right
PageUp	move cursor up by 16 rows
PageDown	move cursor down by 16 rows
Home	move cursor to beginning of pattern
End	move cursor to end of pattern
Shift-Home	move cursor up by exactly one row, overriding Edit Step
Shift-End	move cursor down by exactly one row, overriding Edit Step
Shift-Up	expand selection upwards
Shift-Down	expand selection downwards
Shift-Left	expand selection to the left
Shift-Right	expand selection to the right
Alt-Up	move selection up by one
Alt-Down	move selection down by one
Alt-Left	move selection to previous channel
Alt-Right	move selection to next channel
Backspace	delete note at cursor and/or pull pattern upwards (configurable)
Delete	delete selection
Insert	create blank row at cursor position and push pattern
Ctrl-A	auto-expand selection (select all)
Ctrl-X	cut selection
Ctrl-C	copy selection
Ctrl-V	paste selection
Ctrl-Z	undo
Ctrl-Y	redo
Ctrl-F1	transpose selection (-1 semitone)
Ctrl-F2	transpose selection (+1 semitone)
Ctrl-F3	transpose selection (-1 octave)
Ctrl-F4	transpose selection (+1 octave)
Space	toggle note input (edit)

# effect list

---

some of the effect numbers are taken from ProTracker / FastTracker 2.

however, effects are continuous (unless specified), which means you only need to type it once and then stop it with an effect value of 00 or no effect value at all.

## volume

---

- 0Axy: **Volume slide.**
  - if x is 0 then this slides volume down by y each tick.
  - if y is 0 then this slides volume up by x each tick.
  - FAxy: **Fast volume slide.** same as 0Axy above but 4x faster.
  - F3xx: **Fine volume slide up.** same as 0Ax0 but 64x slower.
  - F4xx: **Fine volume slide down.** same as 0A0x but 64x slower.
  - F8xx: **Single tick volume up.** adds x to volume.
  - F9xx: **Single tick volume down.** subtracts x from volume.
- 
- D3xx: **Volume portamento.** slides the volume to the one specified in the volume column. x is the slide speed.
  - a volume *must* be present with this effect for it to work.
  - D4xx: **Volume portamento (fast).** like D3xx but 4x faster.
- 
- 07xy: **Tremolo.** changes volume to be "wavy" with a sine LFO. x is the speed. y is the depth.
  - tremolo is downward only.
  - maximum tremolo depth is -60 volume steps.
- 
- DCxx: **Delayed mute.** sets channel volume to 0 after xx ticks.

## pitch

---

- E5xx: **Set pitch.** 00 is -1 semitone, 80 is base pitch, FF is nearly +1 semitone.
  - 01xx: **Pitch slide up.**
  - 02xx: **Pitch slide down.**
  - F1xx: **Single tick pitch up.**
  - F2xx: **Single tick pitch down.**
- 
- 03xx: **Portamento.** slides the currently playing note's pitch toward the new note. x is the slide speed.
  - a note *must* be present with this effect for it to work.
  - the effect stops automatically when it reaches the new note.
  - E1xy: **Note slide up.** x is the speed, while y is how many semitones to slide up.
  - E2xy: **Note slide down.** x is the speed, while y is how many semitones to slide down.

- 
- EAx: **Toggle legato**. while on, new notes instantly change the pitch of the currently playing sound instead of starting it over.
  - E6xy: **Quick legato (compatibility)**. transposes note by y semitones after x ticks.
  - if x is between 0 and 7, it transposes up.
  - if x is between 8 and F, it transposes down.
  - E8xy: **Quick legato up**. transposes note up by y semitones after x ticks.
  - E9xy: **Quick legato down**. transposes note down by y semitones after x ticks.
  - 00xy: **Arpeggio**. this effect produces a rapid cycle between the current note, the note plus x semitones and the note plus y semitones.
  - as an example, start with a chord of C-3, G-3, and D#4. the G-3 and D#4 are 7 and 15 semitones higher than the root note, so the corresponding effect is 007F.
  - E0xx: **Set arpeggio speed**. this sets the number of ticks between arpeggio values. default is 1.
- 

- 04xy: **Vibrato**. makes the pitch oscillate. x is the speed, while y is the depth.
- maximum vibrato depth is  $\pm 1$  semitone.
- E3xx: **Set vibrato shape**. xx may be one of the following:
- 00: sine (default)
- 01: sine (upper portion only)
- 02: sine (lower portion only)
- 03: triangle
- 04: ramp up
- 05: ramp down
- 06: square
- 07: random
- 08: square (up)
- 09: square (down)
- 0a: half sine (up)
- 0b: half sine (down)
- E4xx: **Set vibrato range** in 1/16th of a semitone.

## panning

---

not all chips support these effects.

- 08xy: **Set panning**. changes stereo volumes independently. x is the left channel and y is the right one.
  - 88xy: **Set rear panning**. changes rear channel volumes independently. x is the rear left channel and y is the rear right one.
  - 81xx: **Set volume of left channel** (from 00 to FF).
  - 82xx: **Set volume of right channel** (from 00 to FF).
  - 89xx: **Set volume of rear left channel** (from 00 to FF).
  - 8Axx: **Set volume of rear right channel** (from 00 to FF).
- 

- 80xx: **Set panning (linear)**. this effect behaves more like other trackers:
- 00 is left.
- 80 is center.
- FF is right.

- 
- 83xy: **Panning slide.**
  - if y is 0 then this pans to the left by x each tick.
  - if x is 0 then this pans to the right by y each tick.
  - be noted that panning macros override this effect.
  - 84xy: **Panbrello.** makes panning oscillate. x is the speed, while y is the depth.
  - be noted that panning macros override this effect.

## time

---

- 09xx: **Set speed/groove.** if no grooves are defined, this sets speed. if alternating speeds are active, this sets the first speed.
- 0Fxx: **Set speed 2.** during alternating speeds or a groove, this sets the second speed.

- 
- Cxxx: **Set tick rate.** changes tick rate to xxx Hz (ticks per second).
  - xxx may be from 000 to 3FF.
  - F0xx: **Set BPM.** changes tick rate according to beats per minute. range is 01 to FF.

- 
- FDxx: **Set virtual tempo numerator.** sets the virtual tempo's numerator to the effect value.
  - FExx: **Set virtual tempo denominator.** sets the virtual tempo's denominator to the effect value.

- 
- 0Bxx: **Jump to order.** x is the order to play after the current row.
  - this marks the end of a loop with order x as the loop start.
  - 0Dxx: **Jump to next pattern.** skips the current row and remainder of current order. x is the row at which to start playing the next pattern.
  - this can be used to shorten the current order as though it had a different pattern length.
  - FFxx: **Stop song.** stops playback and ends the song. x is ignored.

## note

---

- 0Cxx: **Retrigger.** repeats current note every xx ticks.
- this effect is not continuous; it must be entered on every row.
- ECxx: **Note cut.** triggers note off after xx ticks. this triggers key off in FM/hardware envelope chips, or cuts note otherwise.
- EDxx: **Note delay.** delays note by x ticks.
- FCxx: **Note release.** releases current note after xx ticks. this releases macros and triggers key off in FM/hardware envelope chips.
- E7xx: **Macro release.** releases macros after xx ticks. this does not trigger key off.

## sample offset

---

these effects make the current playing sample on the channel jump to a specific position.  
only some chips support this effect.

sample offset is a 24-bit (3 byte) number.

- 90xx: **Set sample offset (first byte).**
- 91xx: **Set sample offset (second byte).**
- 92xx: **Set sample offset (third byte).**

you may use these effects simultaneously in a row.

if you do not set a byte, its last value will be used.

in previous versions of Furnace a 9xxx effect existed which set the sample position to \$xxx00 (xxx was effectively multiplied by 256). this maps to 920x 91xx in current Furnace.

## other

---

- EBxx: **Set LEGACY sample mode bank.** selects sample bank. used only for compatibility.
- does not apply on Amiga.
- EExx: **Send external command.**
- this effect is currently incomplete.
- F5xx: **Disable macro.**
- F6xx: **Enable macro.**
- F7xx: **Restart macro.**
- see macro table at the end of this document for possible values.

additionally, [each chip has its own effects](#)<sup>221</sup>.

## macro table

---

ID	MACRO
00	volume
01	arpeggio
02	duty/noise
03	waveform
04	pitch
05	extra 1
06	extra 2
07	extra 3
08	extra A (ALG)
09	extra B (FM)
0A	extra C (FMS)
0B	extra D (AMS)
0C	panning left
0D	panning right
0E	phase reset
0F	extra 4

ID	MACRO
10	extra 5
11	extra 6
12	extra 7
13	extra 8
20	AM
21	AR
22	DR
23	MULT
24	RR
25	SL
26	TL
27	DT2
28	RS
29	DT
2A	D2R
2B	SSG-EG
2C	DAM
2D	DVB
2E	EGT
2F	KSL
30	SUS
31	VIB
32	WS
33	KSR
40	<b>operator 2 macros</b>
60	<b>operator 3 macros</b>
80	<b>operator 4 macros</b>

the interpretation of duty, wave and extra macros depends on chip/instrument type:

EX	FM	OPM	OPZ	OPLL	AY-3-8910	AY8930	LYNX	C64
D	NoiseF	NoiseFreq			NoiseFreq	NoiseFreq	Duty/Int	Duty
W		LFO Shape	LFO Shape	Patch	Waveform	Waveform		Waveform
1		AMD	AMD			Duty		FilterMode
2		PMD	PMD		Envelope	Envelope		Resonance
3	LFOSpd	LFO Speed	LFO Speed		AutoEnvNum	AutoEnvNum		Filter Toggle
A	ALG	ALG	ALG		AutoEnvDen	AutoEnvDen		Cutoff



EX	FM	OPM	OPZ	OPLL	AY-3-8910	AY8930	LYNX	C64
B	FB	FB	FB			Noise AND		
C	FMS	FMS	FMS			Noise OR		
D	AMS	AMS	AMS					
4	OpMask	OpMask						Special
5			AMD2					Attack
6			PMD2					Decay
7			LFO2Speed					Sustain
8			LFO2Shape					Release

EX	SAA1099	X1-010	NAMCO 163	FDS	SOUND UNIT	ES5506	MSM6258
D			Wave Pos		Duty	Filt Mode	FreqDiv
W	Waveform	Waveform	Waveform	Waveform	Waveform		
1	Envelope	EnvMode	WaveLen	Mod Depth	Cutoff	Filter K1	ClockDiv
2		Envelope	WaveUpdate	Mod Speed	Resonance	Filter K2	
3		AutoEnvNum	WaveLoad W		Control	Env Count	
A		AutoEnvDen	WaveLoad P			Control	
B			WaveLoad L				
C			WaveLoad T				
D							
4					PResetTime	EnvRampL	
5						EnvRampR	
6						EnvRampK1	
7						EnvRampK2	
8						Env Mode	

EX	QSOUND	SNES	MSM5232	SID2
D	Echo Level	NoiseFreq	GroupCtrl	Duty
W		Waveform		Waveform
1	EchoFeedback	Special	GroupAtk	Filter mode
2	Echo Length	Gain	GroupDec	Resonance
3			Noise	Filter toggle
A				Filter cutoff
B				
C				Noise mode
D				Wave mix mode
4				Special
5				Attack
6				Decay
7				Sustain

EX	Q SOUND	SNES	MSM5232	SID2
8				Release

SID3 instrument also uses some of the FM operators macros in main macros list:

EX	SID3
D	Duty
W	Waveform
1	Special
2	Attack
3	Decay
A	Special wave
B	Phase Mod source
C	Ring Mod source
D	Hard sync source
4	Sustain
5	Sustain rate
6	Release
7	LFSR feedback bits
8	Wave mix mode
OP1 AM	Key On/Off
OP2 AM	Noise phase reset
OP3 AM	Envelope reset
OP4 AM	Noise Arpeggio
OP1 AR	Noise Pitch
OP2 AR	1-bit noise/PCM mode
OP3 AR	Channel signal inversion
OP4 AR	Feedback

SID3 instrument uses FM operators macros for filters:

EX	SID3
D2R	Cutoff
DAM	Resonance
DR	Filter toggle
DT2	Distortion level
DT	Output volume
DVB	Connect to channel input
EGT	Connect to channel output
KSL	Connection matrix row

<b>EX</b>	<b>SID3</b>
KSR	Filter mode

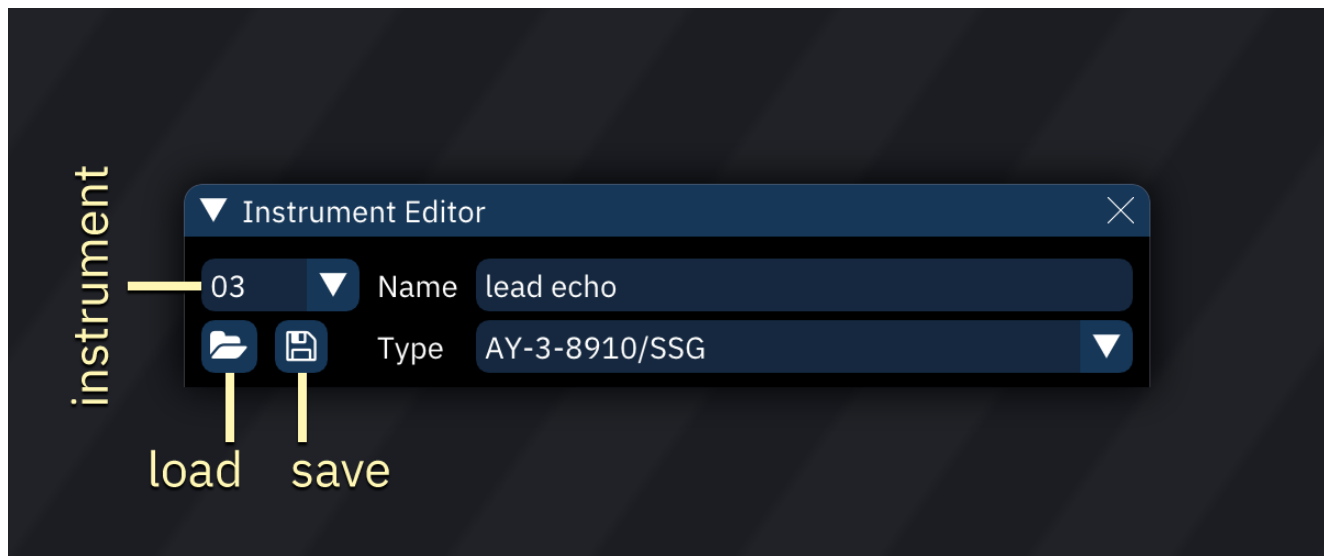
# instrument editor

the instrument editor allows you to edit instruments.

it can be activated by double-clicking on an instrument in the instrument list.

alternatively, window > instrument editor displays it.

## main



- **Select:** displays a list of instruments in the song.
- **Load:** open an instrument file.
- **Save:** save current instrument to a file.
- right-click to see additional options, such as saving in DefleMask preset format (.dmp).
- **Name:** changes the instrument name.
- **Type:** changes the instrument type (usually chip-specific).
- if changed, all applicable settings and macros will remain unchanged.
- you may have to adjust them afterwards.

## instrument types

the following instrument types are available:

- **ADPCM-A**<sub>122</sub> - for use with ADPCM-A sample chip.
- **ADPCM-B**<sub>123</sub> - for use with ADPCM-B sample chip.
- **Atari Lynx**<sub>161</sub> - for use with Atari Lynx handheld console.
- **AY-3-8910/SSG**<sub>124</sub> - for use with AY-3-8910 PSG sound source and SSG portion in YM2610.
- **AY8930**<sub>125</sub> - for use with Microchip AY8930 E-PSG sound source.
- **Beeper**<sub>126</sub> - for use with PC Speaker and ZX Spectrum Beeper (SFX-like engine).
- **Bifurcator**<sub>127</sub> - for use with Bifurcator chip.
- **C140**<sub>128</sub> - for use with C140 sample chip.
- **C219**<sub>129</sub> - for use with C219 sample chip.
- **C64**<sub>130</sub> - for use with Commodore 64 SID.

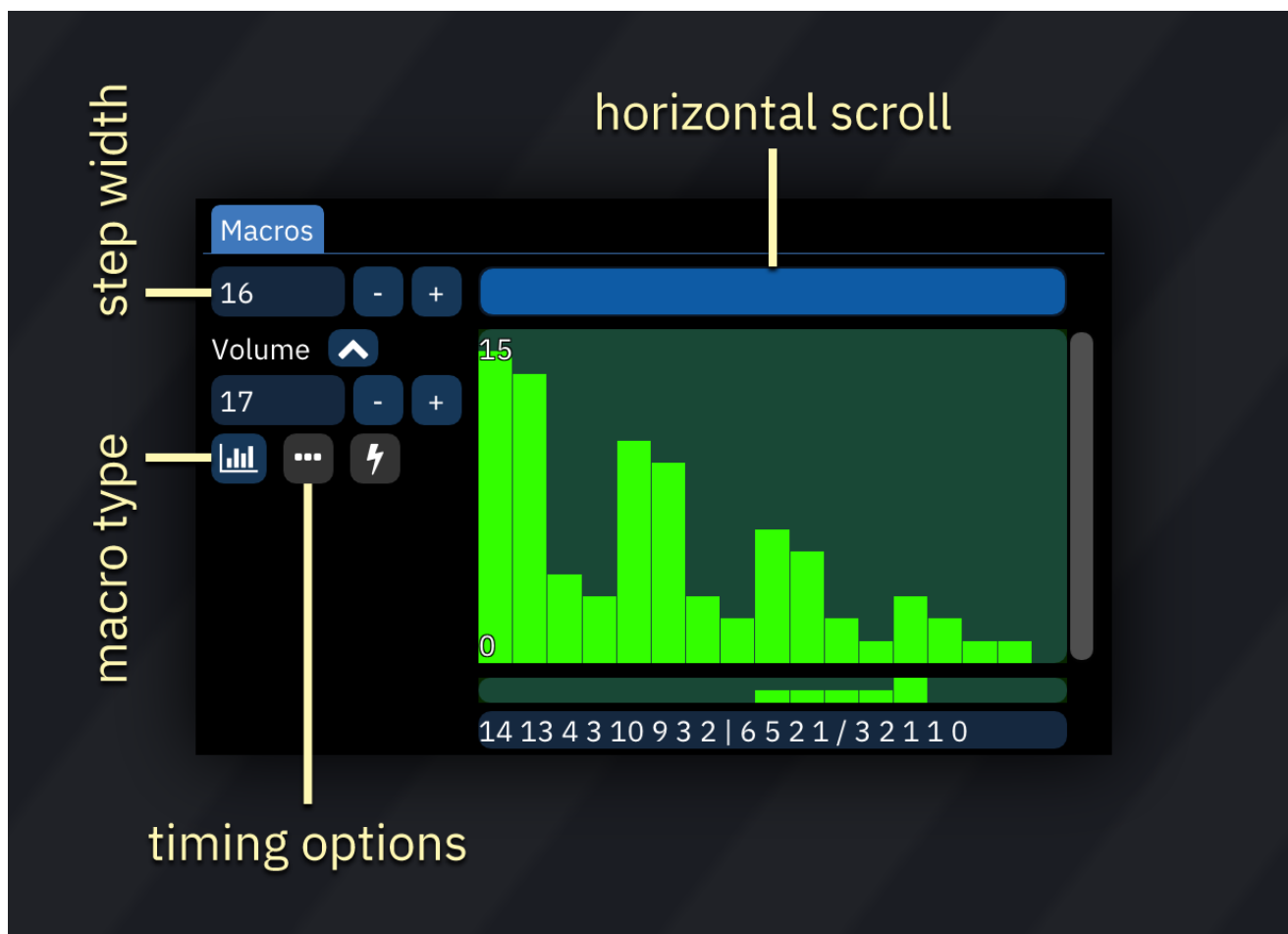
- [Dave](#)<sup>132</sup> - for use with Dave chip.
- [ES5506](#)<sup>133</sup> - for use with Ensoniq ES5506 sound chip.
- [FDS](#)<sup>134</sup> - for use with Famicom Disk System sound source.
- [FM \(ESFM\)](#)<sup>135</sup> - for use with ESFM.
- [FM \(OPL\)](#)<sup>142</sup> - for use with YM3526 (OPL), YM3812 (OPL2) and YMF262 (OPL3).
- [FM \(OPLL\)](#)<sup>142</sup> - for use with YM2413.
- [FM \(OPM\)](#)<sup>145</sup> - for use with YM2151.
- [FM \(OPN\)](#)<sup>148</sup> - for use with YM2612, YM2203, YM2608, YM2610 and YM2610B.
- [FM \(OPZ\)](#)<sup>151</sup> - for use with YM2414.
- [GA20](#)<sup>154</sup> - for use with GA20 sample chip.
- [Game Boy Advance DMA](#)<sup>157</sup> - for use with Game Boy Advance in direct mode.
- [Game Boy Advance MinMod](#)<sup>158</sup> - for use with Game Boy Advance with the MinMod software mixing driver.
- [Game Boy](#)<sup>155</sup> - for use with Game Boy APU.
- [Generic Sample](#)<sup>183</sup> for controlling Amiga and other sample channels/chips like YM2612's Channel 6 PCM mode, NES channel 5, Sega PCM, X1-010 and PC Engine's sample playback mode.
- [K007232](#)<sup>159</sup> - for use with K007232 sample chip.
- [K053260](#)<sup>160</sup> - for use with K053260 sample chip.
- [Konami SCC/Bubble System WSG](#)<sup>184</sup> - for use with Konami SCC and Wavetable portion in Bubble System's sound hardware.
- [MSM5232](#)<sup>163</sup> - for use with MSM5232 PSG sound source.
- [MSM6258](#)<sup>164</sup> - for use with MSM6258 sample chip.
- [MSM6295](#)<sup>165</sup> - for use with MSM6295 sample chip.
- [MultiPCM/OPL4 PCM](#)<sup>166</sup> - for use with OPL4's sample part.
- [Namco 163](#)<sup>168</sup> - for use with Namco 163.
- [Namco WSG](#)<sup>208</sup> - for use with Namco WSG wavetable chips, including C15 and C30.
- [NES](#)<sup>170</sup> - for use with NES.
- [PC Engine](#)<sup>172</sup> - for use with PC Engine's wavetable synthesizer.
- [PET](#)<sup>173</sup> - for use with Commodore PET.
- [Pokémon Mini/QuadTone](#)<sup>174</sup> - for use with these systems.
- [POKEY](#)<sup>175</sup> - for use with Atari 8-bit computers and their POKEY sound source.
- [PowerNoise](#)<sup>176</sup> - for use with PowerNoise chip.
- [PV-1000](#)<sup>179</sup> - for use with Casio PV-1000.
- [QSound](#)<sup>180</sup> - for use with QSound sample chip.
- [RF5C68](#)<sup>181</sup> - for use with RF5C68 sample chip.
- [SAA1099](#)<sup>182</sup> - for use with Philips SAA1099 PSG sound source.
- [SegaPCM](#)<sup>185</sup> - for use with SegaPCM sample chip.
- [Seta/Allumer X1-010](#)<sup>209</sup> - for use with Wavetable portion in Seta/Allumer X1-010.
- [SID2](#)<sup>186</sup> - for use with SID2 fantasy chip.
- [SID3](#)<sup>188</sup> - for use with SID3 fantasy chip.
- [SM8521](#)<sup>192</sup> - for use with SM8521 chip, used in Tiger Game.com.
- [SN76489/Sega PSG](#)<sup>178</sup> - for use with TI SN76489 and derivatives like Sega Master System's PSG.
- [SNES](#)<sup>193</sup> - for use with SNES.
- [Sound Unit](#)<sup>195</sup> - for use with Sound Unit chip.
- [T6W28](#)<sup>197</sup> - for use with Toshiba T6W28 PSG sound source.
- [TED](#)<sup>198</sup> - for use with Commodore Plus/4 and Commodore 16's TED chip.
- [TIA](#)<sup>199</sup> - for use with Atari 2600 chip.
- [VERA](#)<sup>200</sup> - for use with Commander X16 VERA.
- [VIC](#)<sup>201</sup> - for use with VIC-20 sound chip.
- [Virtual Boy](#)<sup>202</sup> - for use with Virtual Boy.
- [VRC6](#)<sup>203</sup> - for use with VRC6's PSG sound source.

- [Watara Supervision](#)<sub>204</sub> - for use with Watara Supervision.
- [WonderSwan](#)<sub>207</sub> - for use with WonderSwan's wavetable synthesizer.
- [YMZ280B](#)<sub>210</sub> - for use with YMZ280B sample chip.

## macros

macros are incredibly versatile tools for automating instrument parameters.

after creating an instrument, open the Instrument Editor and select the "Macros" tab. there may be multiple macro tabs to control individual FM operators and such.






the very first numeric entry sets the visible width of the bars in sequence-type macros. the scroll bar affects the view of all macros at once. there is a matching scrollbar at the bottom underneath all the macros.

each macro has the following parameters:

- macro type (explained below)
- timing options:
  - **Step Length (ticks)**: determines the number of ticks between macro steps. default is 1.
  - **Delay**: delays the macro until this many ticks have elapsed. default is 0.
- the button is highlighted if either of these parameters is set to non-default values.

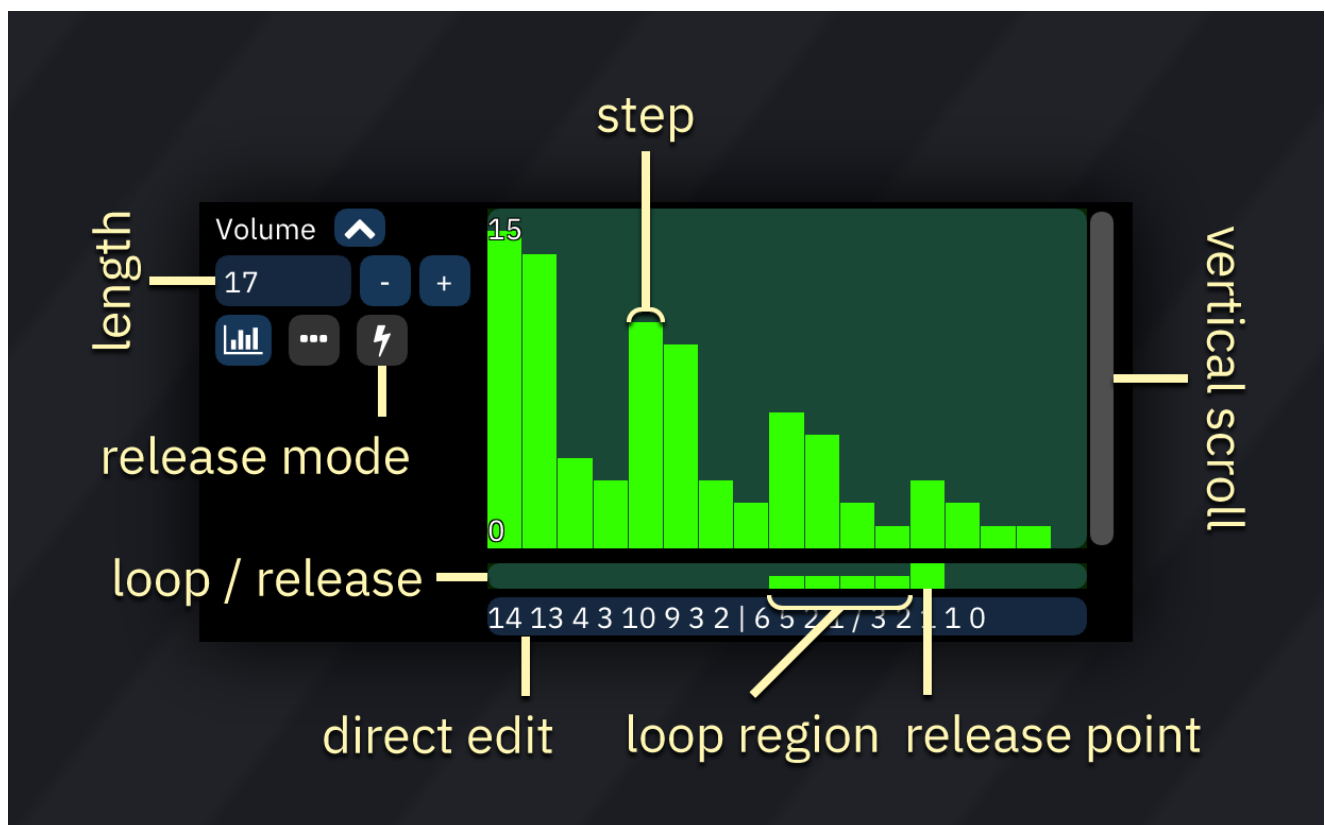
# macro types

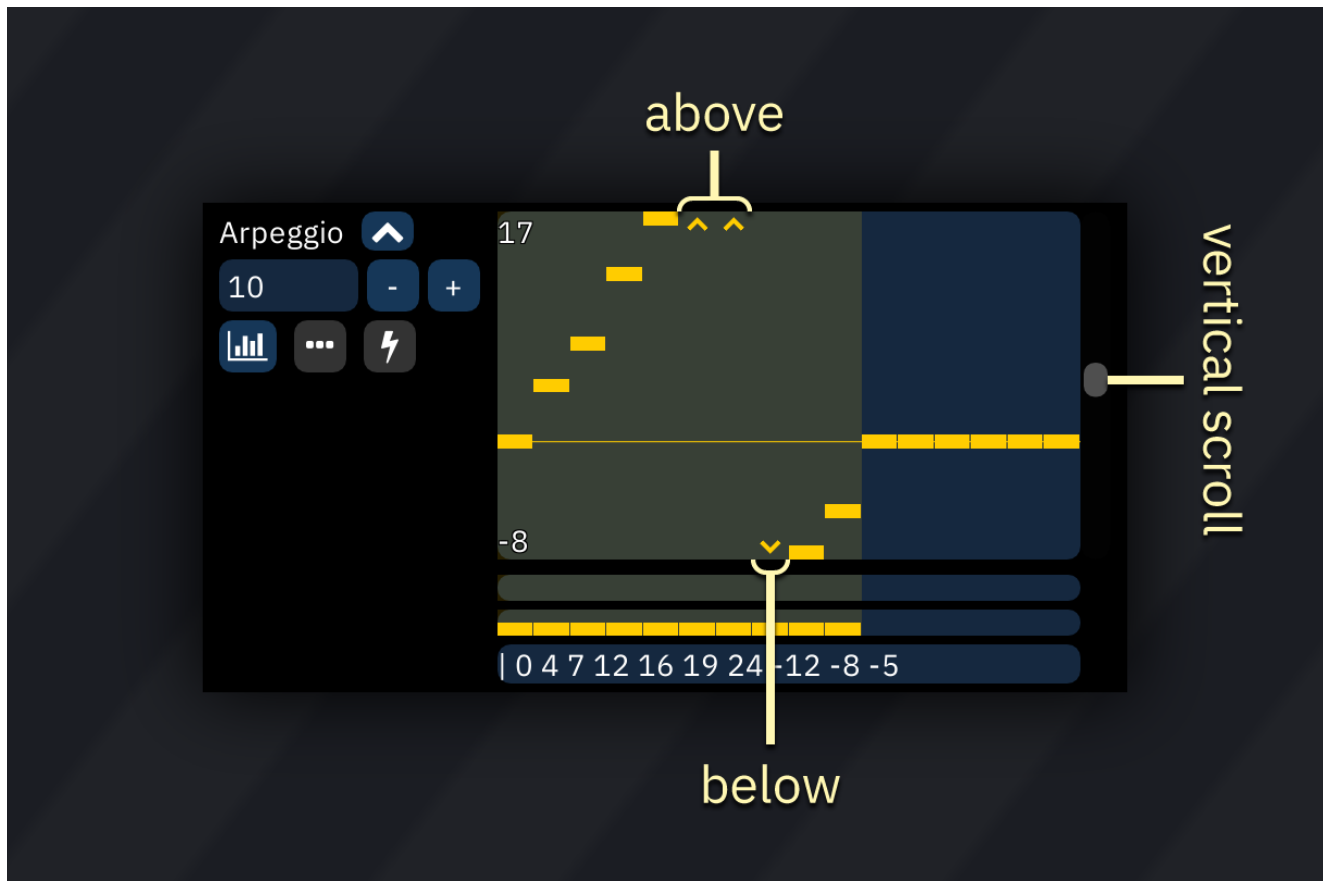
there are three macro types:

-  **Sequence:** a sequence of numeric values.
-  **ADSR:** this is an attack/decay/sustain/release envelope.
-  **LFO:** Low Frequency Oscillator.

## sequence

this is the most basic macro type. when the instrument is played, every value in the macro will be output sequentially, from left to right.





the Length field allows you to set the number of steps in the sequence.

the sequence view allows you to edit the macro.

- press and hold the left mouse button to start drawing. release to stop drawing.
- press and hold the right mouse button to draw a line.
- the start point will be set to the cursor position.
- move the cursor to change the end point.
- release to finish drawing the line.



the sequence view may be adjusted using the following combinations:

- hold Ctrl and use the scroll wheel to zoom horizontally.
- hold Ctrl-Shift and use the scroll wheel to zoom vertically.
- the scrollbar at the right allows you to scroll vertically (if possible).
- you may hold Shift and use the scroll wheel to scroll vertically as well.

right-click on the sequence view to open a menu:

- **copy**: copy this macro to clipboard.
- **paste**: pastes the macro.
- **clear**: clears the macro.
- **clear contents**: resets all values to 0.
- **offset**:
- **X**: slides the data "horizontally" within the macro, filling the gap with zeroes. data moved past the start or end is lost.
- **Y**: increases or decreases all values, clipping them if they would move past the allowed range.
- **scale**:
- **X**: stretches the macro.
- **Y**: multiplies all values by the scale factor, clipping them if they would exceed the allowed range.
- **randomize**: replaces all values with random values between **Min** and **Max**.

arpeggio and pitch macros may have values above or below the visible area. indicators will be shown until they are scrolled into view.

bitmask-style macros show labels for each of their bits. these are edited as toggles.

- drawing lines is not possible in these macros.

under the sequence view there is a bar that allows you to set loop and release points.

- click to set the loop start point; the end point is the last step or release point.
- right-click to remove the loop point.
- shift-click to set the release point.
- the macro will stop at the release point until the note is released (=== or REL).
  - if the loop point is set, and it is placed before the release point, the macro will loop until note release instead.
- shift-right-click to remove the release point.

arpeggio macros have an additional bar under the sequence view to set steps to "relative" or "fixed":

- by default, step values are offsets **relative** to the note.
- if clicked on, a step value becomes **fixed** and will be played at its corresponding note without regard to the currently playing note.
- values are counted from C-0. for example, a fixed value of 48 produces a C-4 note.
- fixed values are especially useful for noise or percussion.

the sequence can be edited in the text input field at the very bottom. the following symbols have special meanings:

- |: loop point.
- /: release point.

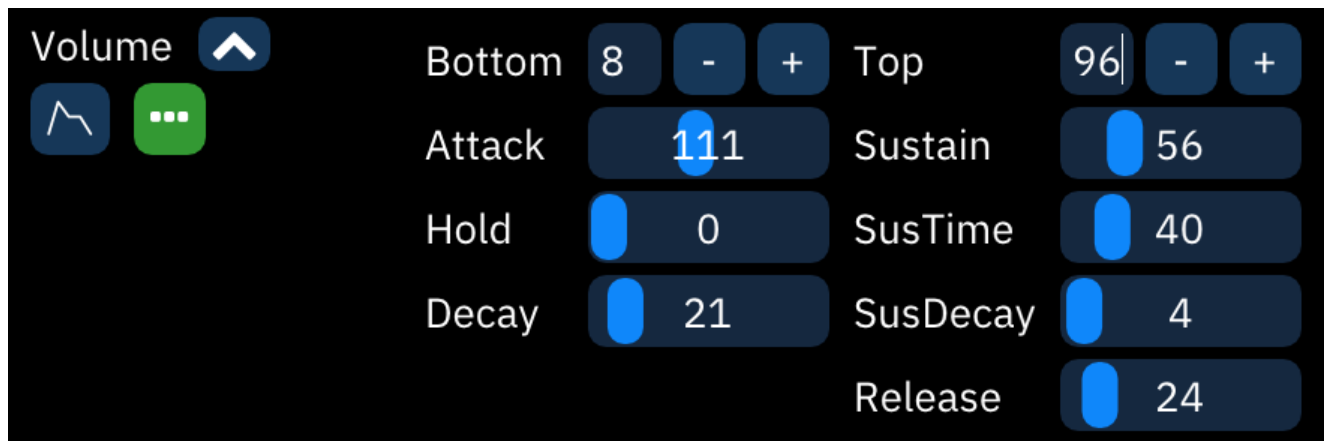
- in arpeggio macros, @ prefixed to a value indicates that it is a fixed value as described above.

in bitmask-style macros, the values are added up in binary and converted to decimal.

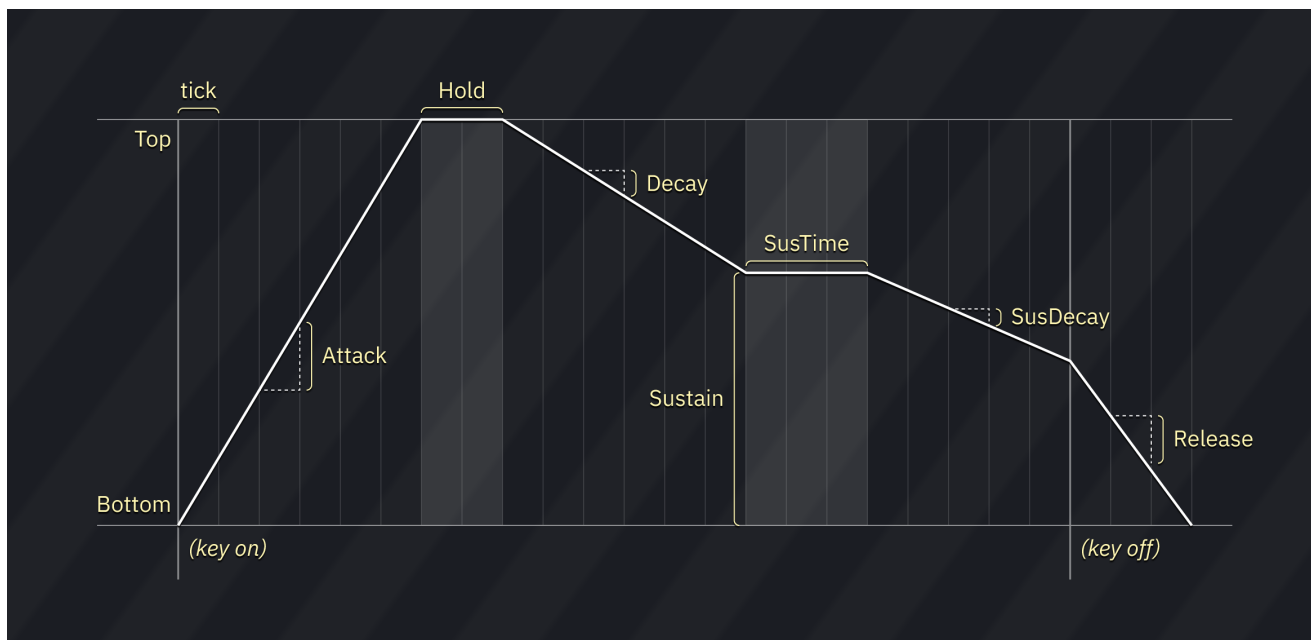
the release mode parameter determines how macro release (=== or REL in the pattern) is handled:

- **Active:** jumps to release position on release.
- **Passive:** does not jump to release position. this will result in delay if release position has not been reached yet.

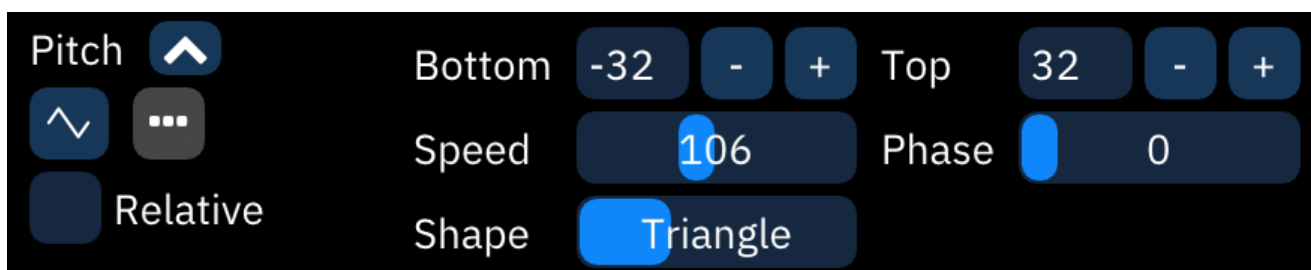
## ADSR



- **Bottom** and **Top** determine the macro's output range (Bottom can be larger than Top to invert the envelope!). all outputs will be between these two values.
- Attack, Decay, Sustain, SusDecay, and Release accept inputs between 0 to 255. these are scaled to the distance between Bottom and Top.
- the output starts at Bottom.
- **Attack** is how much the output moves toward Top with each tick.
- **Hold** sets how many ticks to stay at Top before Decay.
- **Decay** is how much the output moves to the Sustain level.
- **Sustain** is how far from Bottom the value stays while the note is on.
- **SusTime** is how many ticks to stay at Sustain until SusDecay.
- **SusDecay** is how much the output moves toward Bottom with each tick while the note is on.
- **Release** is how much the output moves toward Bottom with each tick after the note is released.



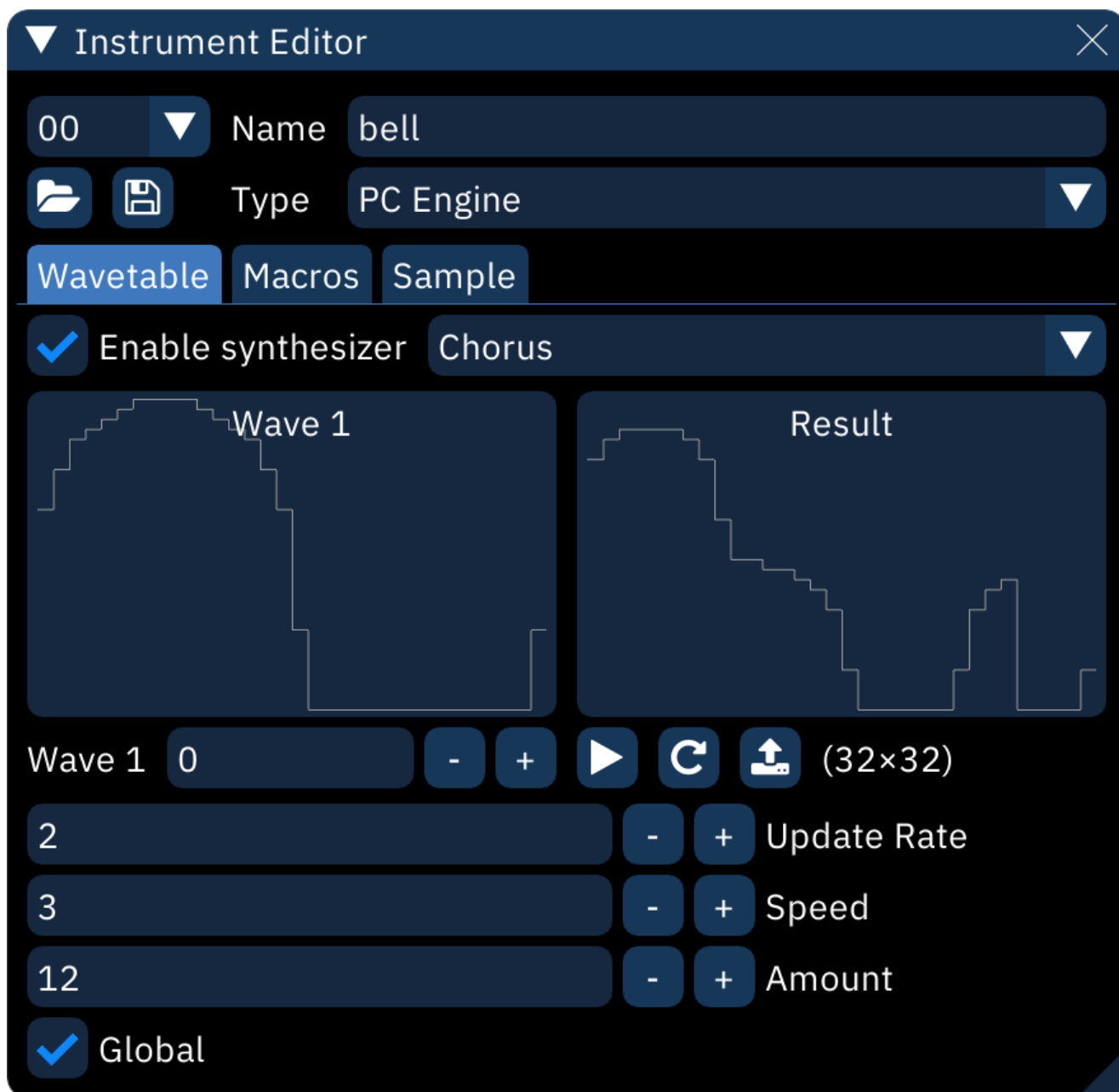
## LFO



- **Bottom** and **Top** determine the macro's output range (Bottom can be larger than Top to invert the waveform!).
- **Speed** is how quickly the LFO position moves.
- **Phase** defines the starting LFO position, measured in 1/1024 increments.
- **Shape** is the waveform of the LFO. there are three waveforms:
  - Triangle: Bottom > Top > Bottom.
  - Saw: moves from Bottom to Top, and then jumps back to Bottom.
  - Square: alternates between Bottom and Top.

## wavetable

this tab appears for PC Engine, FDS, Namco WSG, and other wavetable-based instruments.



when **Enable synthesizer** is off, the wavetable used for the instrument may be selected by creating a Waveform macro with a single value.

to use the wavetable synthesizer, refer to [the wavetable synthesizer section](#)<sup>205</sup>.

## sample

this tab appears for Generic PCM DAC, Amiga and SNES.

Instrument Editor

OF

Name

percussion

Type

SNES

Sample

SNES

Wavetable

Macros

Sample

Tambourine DQVI

☐

Use wavetable

☒

Use sample map

	#	note	sample name
B-2	---	B-2	---
C-3	0	C-3	DrumBass 0
C#3	2	C#3	DrumSnare 0
D-3	1	D-3	DrumRim 0
D#3	3	F-4	DrumRide 0
E-3	8	C-2	Tambourine DQVI
F-3	---	F-3	---
F#3		F#3	

see the [Generic Sample section](#)<sup>183</sup> for more information.

# ADPCM-A instrument editor

---

the ADPCM-A instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

note that samples on ADPCM-A are tied to one frequency.

## Macros

---

- **Volume:** volume sequence.
- **Global Volume:** sets the global volume of the ADPCM-A part.
- **Panning:** toggle left/right output.
- **Phase Reset:** trigger restart of sample.

# ADPCM-B instrument editor

---

the ADPCM-B instrument editor contains three tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Panning:** toggle left/right output.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of sample.

# AY-3-8910 instrument editor

---

the AY-3-8910 instrument editor consists of three tabs.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

note that using samples on AY is CPU expensive!

## Timer Macros

---

- **Timer FX:** timer effect type sequence. 0- off, 1- tone PWM, 2- envelope distortion, 3- reserved
- **TFX Offset:** PWM speed sequence.
- **Timer Num and Den:** multiplier of a virtual square wave modulator (?)
- **PWM Boundary:** sets the range of a pulse width modulation

note that timer effects might not be supported by VGM players, and are rather CPU expensive!

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Noise Freq:** noise generator frequency sequence.
- note: global!
- **Waveform:** selector of sound type - square wave tone, noise and/or envelope generator.
- you may select more than one option.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of **envelope**.
- **Envelope:** configures the hardware envelope.
- **enable:** sets whether envelope is enabled.
- **direction:** flips the envelope's direction.
- **alternate:** when enabled, the envelope will change direction once it hits a boundary.
- **hold:** sets whether the envelope should stop when hitting boundary, or loop.
- **AutoEnv Num:** sets the envelope to the channel's frequency multiplied by numerator.
- **AutoEnv Den:** sets the envelope to the channel's frequency multiplied by denominator.
- these two must be set in order for AutoEnv to work!
- **Force Period:** sets the tone period (wavelength).
- overrides Arpeggio and Pitch macros.
- **Env Period:** sets the envelope period.
- ignored if both AutoEnv macros are set.



# AY8930 instrument editor

---

the AY8930 instrument editor consists of two tabs.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

note that using samples on AY is CPU expensive!

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Noise Freq:** noise generator frequency sequence.
- **note:** global!
- **Waveform:** selector of sound type - square wave tone, noise and/or envelope generator.
- you may select more than one option.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of **envelope**.
- **Duty:** duty cycle sequence.
- **Envelope:** configures the hardware envelope.
- **enable:** sets whether envelope is enabled.
- **direction:** flips the envelope's direction.
- **alternate:** when enabled, the envelope will change direction once it hits a boundary.
- **hold:** sets whether the envelope should stop when hitting boundary, or loop.
- **AutoEnv Num:** sets the envelope to the channel's frequency multiplied by numerator.
- **AutoEnv Den:** sets the envelope to the channel's frequency multiplied by denominator.
- these two must be set in order for AutoEnv to work!
- **Force Period:** sets the tone period (wavelength).
- overrides Arpeggio and Pitch macros.
- **Env Period:** sets the envelope period.
- ignored if both AutoEnv macros are set.
- **Noise AND Mask:** alters the shape/frequency of the noise generator, allowing to produce various interesting sound effects and even PWM phasing.
- **Noise OR Mask:** see above.

# beeper instrument editor

---

used in PC Speaker and ZX Spectrum (SFX-like engine).

- **Volume:** on-off volume sequence.
- **Arpeggio:** pitch sequence.
- **Pulse Width:** pulse width sequence.
- only on ZX Spectrum.
- **Pitch:** fine pitch.

# Bifurcator instrument editor

---

Bifurcator instrument editor consists of these macros:

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Parameter**: set parameter of logistic map function.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Load Value**: changes the current output value.

## audio generation description

---

Bifurcator uses logistic map iterations for sound generation.  
basically it runs the following function over and over:

```
r = (1 + (p / 65536)) * 2  
x = (r * x) * (1 - x)
```

where x is the current output value and p is the "parameter".

by varying the parameter, the value of x may change drastically, producing a variety of sounds.  
the higher the parameter, the more "chaos" is present, effectively yielding noise.

the default parameter is 47360, which results in a square wave.

if the parameter is set to 0, there's no sound at all.

as the parameter approaches 32768, a decaying square wave is produced.

the square wave stops decaying past 32768 and becomes louder until the parameter hits ~47496  
( $r = 1 + \sqrt{6}$ ).

a second square wave one octave lower then starts appearing, until the parameter reaches ~51443  
( $r \approx 3.56995$ ). this is where chaos begins.

anything higher results in a total mess.

however, at ~59914 ( $r = 1 + \sqrt{8}$ ) you can hear a 33% pulse wave.

## the importance of loading the value

---

you must load a value that isn't 0 in order to get sound. otherwise the function will always output 0.

# Namco C140 instrument editor

---

the Namco C140 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of sample.

# Namco C219 instrument editor

---

the Namco C219 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Control**: channel control sequence:
- **surround**: invert the right output for a surround effect.
- **invert**: invert both outputs. when used together with surround, this inverts just the left output.
- **noise**: toggles noise mode.
- setting control bits restart the sample!
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.

# C64 SID instrument editor

---

the C64 instrument editor consists of two tabs: "C64" to control various parameters of sound channels, and "Macros" containing several macros.

## C64

---

- **Waveform:** allows selecting a waveform.
- more than one waveform can be selected at once. in that case a logical AND mix of waves will occur...
  - due to hardware flaws, the mixing is a bit weird and sounds different between the 6581 and the 8580.
- noise is an exception. it cannot be used with any of the other waveforms.
- **Attack:** determines the rising time for the sound. the bigger the value, the slower the attack. (0 to 15).
- **Decay:** determines the diminishing time for the sound. the higher the value, the longer the decay (0 to 15).
- **Sustain:** sets the volume level at which the sound stops decaying and holds steady (0 to 15).
- **Release:** determines the rate at which the sound fades out after note off. the higher the value, the longer the release (0 to 15).
- **Duty:** specifies the width of a pulse wave (0 to 4095).
- **Reset duty on new note:** overwrite current duty value with the one that is specified in the instrument on new note.
- only useful when using relative duty macro.
- **Ring Modulation:** when enabled, the channel's output will be multiplied with the previous channel's.
- **Oscillator Sync:** enables oscillator hard sync. as the previous channel's oscillator finishes a cycle, it resets the period of the channel's oscillator, forcing the latter to have the same base frequency. this can produce a harmonically rich sound, the timbre of which can be altered by varying the synchronized oscillator's frequency.
- **Enable filter:** when enabled, this instrument will go through the filter.
- **Initialize filter:** initializes the filter with the specified parameters:
- **Cutoff:** the filter's point in where frequencies are cut off (0 to 2047).
- **Resonance:** amplifies or focuses on the cutoff frequency, creating a secondary peak forms and colors the original pitch (0 to 15).
- **Filter mode:** sets the filter mode. you may pick one or more of the following:
  - **low:** a low-pass filter. the lower the cutoff, the darker the sound.
  - **high:** a high-pass filter. higher cutoff values result in a less "bassy" sound.
  - **band:** a band-pass filter. cutoff determines which part of the sound is heard (from bass to treble).
  - **ch3off:** mutes channel 3 when enabled. it was originally planned for usage with two registers where program could read current oscillator and envelope outputs, thus making vibrato and SFX generation easier. but who wanted to sacrifice one channel out of three! so aforementioned was just done in software, and the feature was never used.
  - multiple filter modes can be selected simultaneously. for example, selecting both "low" and "high" results in a bandstop (notch) filter.
- **Absolute Cutoff Macro:** when enabled, the cutoff macro will go from 0 to 2047, and it will be absolute (in other words, control the cutoff directly rather than being relative).

- **Absolute Duty Macro:** when enabled, the duty macro will go from 0 to 4095.
- **Don't test before new note:** this option disables the one-tick hard reset and test bit before a new note.

## Macros

---

- **Volume:** volume sequence.
- warning: volume sequence is global! this means it controls the chip's volume and therefore affects all channels.
- **Arpeggio:** pitch sequence.
- **Duty:** pulse width sequence.
- **Waveform:** select the waveform used by instrument.
- **Pitch:** fine pitch.
- **Cutoff:** filter cutoff.
- **Filter mode:** select the filter mode.
- **Resonance:** filter resonance sequence.
- **Special:** ring and oscillator sync selector, as well as:
- **gate bit:**
  - set (1): key on. if previous state was 0 it triggers envelope start/restart; if previous state was 1, it does nothing.
  - reset (0): key off. if previous state was 1 it triggers envelope release; if previous state was 0, it does nothing.
- **test bit:**
  - set (1): immediately mute channel
  - if the channel is a source of ring mod and/or hard sync, those stop working until the bit is reset.
  - reset (0): unmute channel and restore ring mod/hard sync.
- **Attack:** sets envelope attack speed.
- if you modify attack speed when the envelope is in attack phase it immediately changes.
- **Decay:** sets envelope decay speed.
- if you modify decay speed when envelope is in decay phase it immediately changes.
- **Sustain:** sets envelope sustain level.
- if you modify sustain level when envelope is in sustain phase it immediately changes, although you can only go down. for example, 9-to-8 and 8-to-8 both work, but 8-to-9 immediately mutes the channel.
- **Release:** sets envelope release speed.
- if you modify release speed when envelope is in release phase it immediately changes.

# Dave instrument editor

---

the Dave instrument editor consists of these macros:

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Noise Freq**: set noise frequency source.
  - 0: fixed frequency (~62.5KHz)
  - 1: channel 1
  - 2: channel 2
  - 3: channel 3
- **Waveform**: select waveform or noise length.
  - 0: square
  - 1: bass
  - 2: buzz
  - 3: reed
  - 4: noise
- for noise channel the range is 0 to 3.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform.
  - does not apply for noise channel.
- **Control**: set channel parameters.
  - **low pass (noise)**: enable low-pass filter. only in noise channel.
  - **swap counters (noise)**: enable swap counters mode. only in noise channel.
  - **ring mod**: enable ring mod with channel+2.
  - **high pass**: enable high-pass filter with the next channel.



# Ensoniq ES5506 instrument editor

---

the ES5506 instrument editor contains three tabs: Sample, ES5506 and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## ES5506

---

ES5506 contains a filter, which is somewhat configurable.  
there's also a hardware envelope, but it's probably most useful for smoothing.

you may use this tab to set up ES5506-specific parameters:

- **Filter Mode:** sets filter mode.
- **HP/K2, HP/K2:** run high-pass twice using filter K2.
- **HP/K2, LP/K1:** run high-pass using filter K2, and then low-pass using filter K1.
- **LP/K2, LP/K2:** run low-pass twice using filter K2.
- **LP/K2, LP/K1:** run low-pass using filter K2, and then again using filter K1.
- **Filter K1:** set coefficient 1 (K1). effectively controls cutoff.
- **Filter K2:** set coefficient 2 (K2). effectively controls cutoff.
- **Envelope count:** set length of hardware envelope (it's very short even at highest value).
- **Left Volume Ramp:** how much to change left volume on every envelope step.
- **Right Volume Ramp:** how much to change right volume on every envelope step.
- **Filter K1 Ramp:** how much to change filter K1 every envelope step.
- **Filter K2 Ramp:** how much to change filter K2 on every envelope step.
- **K1 Ramp Slowdown:** increases length of K1 ramp.
- **K2 Ramp Slowdown:** increases length of K2 ramp.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Filter Mode:** sets filter mode.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.
- **Filter K1:** K1 sequence.
- **Filter K2:** K2 sequence.
- **Outputs:** ES5506 has 6 stereo output (total 12). select which pair of outputs to use.
- **Control:** reverse/pause sequence.

# FDS instrument editor

---

the FDS instrument editor contains three tabs: FDS, Wavetable and Macros.

## FDS

---

here you can edit FDS-specific settings.

- **Compatibility mode:** DO NOT ENABLE. this exists for DefleMask compatibility. leave it alone.
- **Modulation depth:** sets frequency modulation depth.
- **Modulation speed:** sets frequency modulation speed.
- **Modulation table:** this allows you to define a waveform for frequency modulation.
  - the range is -4 to 3.
  - a value of -4 will reset the modulator.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Waveform:** wavetable sequence.
- **Pitch:** fine pitch.
- **Mod Depth:** modulation depth.
- **Mod Speed:** modulation speed.
- **Mod Position:** sets position of modulator.

# ESFM instrument editor

---

the ESFM editor is divided into 6 tabs:

- **FM**: for controlling the basic parameters of FM sound source.
- **Macros (OP1)**: for macros controlling FM parameters of operator 1.
- **Macros (OP2)**: for macros controlling FM parameters of operator 2.
- **Macros (OP3)**: for macros controlling FM parameters of operator 3.
- **Macros (OP4)**: for macros controlling FM parameters of operator 4.
- **Macros**: for other macros (volume/arp/pitch/pan/operator 4 noise mode).

## FM

---

ESFM is four-operator, meaning it takes four oscillators to produce a single sound.

unlike most four-operator FM synthesizers, however, ESFM does not have an algorithm selection. instead, it uses a fixed operator arrangement, but allows you to independently control the output and modulation input levels of each operator. this allows it to reproduce a few common four-operator algorithms, as well as unique combinations where operators act as modulators and carriers at the same time.

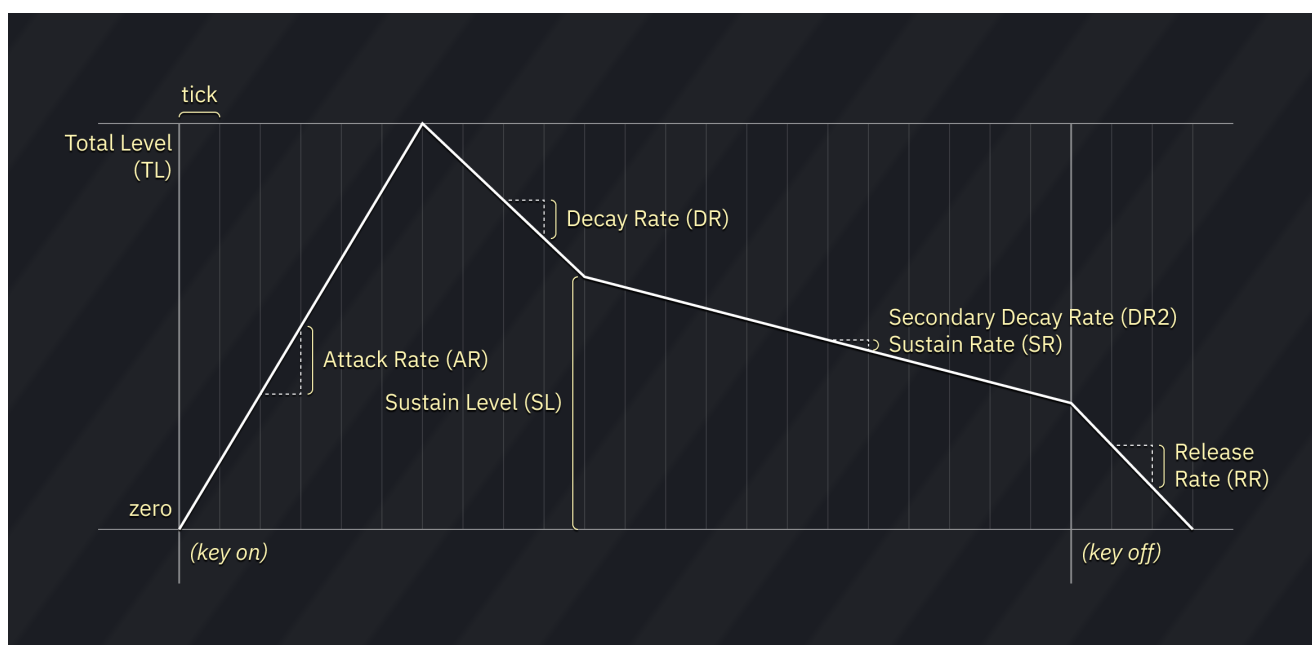
these apply to the instrument as a whole:

- **OP4 Noise Mode**: determines the mode used to produce noise in operator 4.
- Normal: noise is disabled.
- Snare: takes the snare noise generation mode from OPL. square + noise.
- HiHat: ring modulates with operator 3 and adds noise.
- Top: ring modulates with operator 3 and double pitch modulation input.
- these are normally used for the drum channels in ESFM's OPL3 compatibility mode.
  - however, in ESFM, operator 4 can be modulated by operator 3, whereas in OPL3 drum mode the rhythm channels cannot be modulated.
- **note**: usage of noise mode "Top" is discouraged for now as it is not properly emulated yet, and results may change when the emulation gets fixed in the future.
- **Octave**: sets the "block" of the frequency register, which affects note range and precision.
- **operator routing preview**: shows how operators are connected with each other and with the audio output (at the bottom).
- left-click pops up a small "operators changes with volume?" dialog where each operator can be toggled to scale with volume level.
- right-click switches to a preview display of the waveform generated on a new note:
  - left-click restarts the preview.
  - middle-click pauses and unpauses the preview.
  - right-click returns to algorithm view.

these apply to each operator:

- the crossed-arrows button can be dragged to rearrange operators.
- **Amplitude Modulation (AM)**: makes the operator affected by LFO tremolo.
- **AM Depth (DAM/AMD)**: when enabled, LFO tremolo is deeper (1dB off; 4.8dB on).

- **Sustain flag (SUS)**: when enabled, the envelope pauses ("sustains") once it reaches the Sustain Level and does not proceed to the release phase until note off.
- **Envelope Delay (DL)**: determines the delay time before the envelope is triggered. the bigger the value, the longer the delay (0 to 7).
  - a change of one unit doubles or halves the delay time.
  - a value of 0 results in no delay.
- **Attack Rate (AR)**: determines the rising time for the sound. the bigger the value, the faster the attack (0 to 15).
- **Decay Rate (DR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. it's the initial amplitude decay rate (0 to 15).
- **Sustain Level (SL)**: determines the point at which the sound ceases to decay and changes to a sound having a constant level. the sustain level is expressed as a fraction of the maximum level (0 to 15).
- **Release Rate (RR)**: determines the rate at which the sound disappears after note off. the higher the value, the shorter the release (0 to 15).
- **Total Level (TL)**: represents the envelope's highest amplitude, with 0 being the largest and 63 (decimal) the smallest. a change of one unit is about 0.75 dB.
- **Key Scale Level (KSL)**: also known as "Level Scale". determines the degree to which the amplitude decreases according to the pitch.



- **Key Scale Rate (KSR)**: also known as "Rate Scale". determines the degree to which the envelope execution speed increases according to the pitch.
- **Frequency Multiplier (MULT)**: sets the coarse pitch offset in relation to the note (0 to 15). the values follow the harmonic scale. for example, 0 is -1 octave, 1 is 0 octaves, 2 is 1 octave, 3 is 1 octave 7 semitones, and so on.
  - note that values 11, 13 and 14 behave as 10, 12 and 15 respectively.
- **Tune (CT)**: sets the semitone offset in relation to the note (-24 to 24).
  - this is a software effect.
- **Fine Detune (DT)**: shifts the pitch in fine steps (-128 to 127). 0 is the base pitch, -128 is -1 semitone, 127 is nearly +1 semitone.
  - this is a software effect.
- **Left (L)**: toggles output to the left channel from the operator to the audio output.
- **Right (R)**: toggles output to the right channel from the operator to the audio output.

- **Waveform Select (WS)**: changes the waveform of the operator (0 to 7).
- **Vibrato (VIB)**: makes the operator affected by LFO vibrato.
- **Vibrato Depth (DVB/FMD)**: when enabled, vibrato is deeper.

## routing controls

- **Output Level (OL)**: sets the output level from this operator to the audio output (0 to 7).
- 7 is the loudest level and 1 is the softest, while 0 disables audio output.
- a change of one unit is about 6 dB.
- this output scaling factor is applied after TL and envelope scaling have been performed.
- **Modulation Input Level (MI)**: sets the modulation level from the previous operator to this operator (0 to 7).
- 7 is the strongest level and 1 is the weakest, while 0 disables modulation.
- a change of one unit is about 6 dB.
- for operator 1 this controls the **feedback level**.
- this modulation scaling factor is applied after the previous operator's TL and envelope scaling have been performed, but is unaffected by OL above.

## common algorithms

this table contains a list of modulation input/output level values which resemble common algorithms in Yamaha FM chips.

**note:** MI1 is not included as it is the feedback level.

ALGORITHM	OL1	MI2	OL2	MI3	OL3	MI4	OL4
OPN algorithm 0	0	7	0	7	0	7	7
OPN algorithm 4	0	7	7	0	0	7	7
OPN algorithm 6	0	7	7	0	7	0	7
OPN algorithm 7	7	0	7	0	7	0	7
OPL3 algorithm 1	7	0	0	7	0	7	7
OPL3 algorithm 3	7	0	0	7	7	0	7
OPL3 algorithm 1 (variant)	0	7	0	7	7	0	7
#### fixed frequency mode							

each operator has a Fixed Frequency mode. once enabled, the operator runs at the specified frequency regardless of the note.

when fixed frequency mode is enabled, the Tune and Fine Detune sliders will be replaced by **Block (Blk)** and **FreqNum (F)**, which allow you to input a raw frequency value into the operator. the actual frequency is determined by the formula:  $\text{FreqNum} \times (2^{\text{Block}})$ .

in other words, FreqNum defines the base frequency, while Block determines the scaling factor in octaves.

## macros

these macros allow you to control several parameters of FM per tick.

# OP1-OP4 Macros

---

most parameters are listed above.

## envelope delay macro tricks

due to a quirk in how the envelope delay feature works, the **Envelope Delay** macro can control the operator's key-on status for a limited amount of time after a note is played. a value of 0 represents key-on, while a value of 7 represents key-off.

note that the macro cannot exceed 659.1 ms in length - anything beyond that will be treated as a value of 0.

## operator arpeggio and pitch macros

among the available macros are **Op. Arpeggio** and **Op. Pitch**. these work like the **Arpeggio** and **Pitch** macros featured below, but are applied to the individual operator, overriding the **Arpeggio**/**Pitch** macros respectively.

the **Tune** and **Fine Detune** FM parameters are still respected when using these macros.

## fixed frequency macros

when fixed frequency is enabled for an operator, the **Op. Arpeggio** and **Op. Pitch** macros will be replaced by the **Block** and **FreqNum** macros. these can be used to change the operator's fixed frequency over time.

# Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **OP4 Noise Mode**: operator 4 noise mode sequence.
- **Panning**: enables output on left/right channels.
- note that each operator also has its own pan controls, which get masked by this global pan control.
- **Pitch**: fine pitch.
- **Relative**: when enabled, pitch changes are relative to the current pitch.
- **Phase Reset**: restarts all operators and resets the waveform to its start.

# OPL FM synthesis instrument editor

---

the OPL FM editor is divided into 7 tabs:

- **FM**: for controlling the basic parameters of FM sound source.
- **Macros (FM)**: for macros controlling algorithm and feedback.
- **Macros (OP1)**: for macros controlling FM parameters of operator 1.
- **Macros (OP2)**: for macros controlling FM parameters of operator 2.
- **Macros (OP3)**: for macros controlling FM parameters of operator 3 (only when 4-op flag is set and only on OPL3!).
- **Macros (OP4)**: for macros controlling FM parameters of operator 4 (only when 4-op flag is set and only on OPL3!).
- **Macros**: for other macros (volume/arp/pitch/pan).

## FM

---

the OPL synthesizers are nominally two-operator (OPL3 supports 4-operator mode on up to six channels), meaning it takes two oscillators to produce a single sound.

these apply to the instrument as a whole:

- **Algorithm (ALG)**: determines how operators are connected to each other (0-1 range and OPL1 and OPL2; 0-3 range on OPL3 4op mode).
- left-click pops up a small "operators changes with volume?" dialog where each operator can be toggled to scale with volume level.
- right-click to switch to a preview display of the waveform generated on a new note:
  - left-click restarts the preview.
  - middle-click pauses and unpauses the preview.
  - right-click returns to algorithm view.
- **Feedback (FB)**: determines how many times operator 1 returns its output to itself (0 to 7).
- **Octave**: sets the "block" of the frequency register, which affects note range and precision.
- **4-op**: enables 4-operator FM instrument editor mode (only on OPL3).
- **Drums**: enables OPL drum mode editor.

these apply to each operator:

- the crossed-arrows button can be dragged to rearrange operators.
- **Amplitude Modulation (AM)**: makes the operator affected by LFO tremolo.
- **Sustain flag (SUS)**: when enabled, the envelope pauses ("sustains") once it reaches the Sustain Level and does not proceed to the release phase until note off.
- **Attack Rate (AR)**: determines the rising time for the sound. the bigger the value, the faster the attack (0 to 15).
- **Decay Rate (DR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. it's the initial amplitude decay rate (0 to 15).

- **Sustain Level (SL)**: determines the point at which the sound ceases to decay and changes to a sound having a constant level. the sustain level is expressed as a fraction of the maximum level (0 to 15).
- **Release Rate (RR)**: determines the rate at which the sound disappears after note off. the higher the value, the shorter the release (0 to 15).
- **Total Level (TL)**: represents the envelope's highest amplitude, with 0 being the largest and 63 (decimal) the smallest. a change of one unit is about 0.75 dB.
- **Key Scale Level (KSL)**: also known as "Level Scale". determines the degree to which the amplitude decreases according to the pitch.



- **Key Scale Rate (KSR)**: also known as "Rate Scale". determines the degree to which the envelope execution speed increases according to the pitch.
- **Frequency Multiplier (MULT)**: sets the coarse pitch offset in relation to the note (0 to 15). the values follow the harmonic scale. for example, 0 is -1 octave, 1 is 0 octaves, 2 is 1 octave, 3 is 1 octave 7 semitones, and so on.
- note that values 11, 13 and 14 behave as 10, 12 and 12 respectively.
- **Waveform Select (WS)**: changes the waveform of the operator (OPL2 and OPL3 only, 0-3 range on OPL2 and 0-7 on OPL3).
- **Vibrato (VIB)**: makes the operator affected by LFO vibrato.

## macros

these macros allow you to control several parameters of FM per tick.

## FM Macros

all parameters are listed above.

## OP1-OP4 Macros

all parameters are listed above.



# Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Panning:** enables output on left/right/rear channels. OPL3 only.
- **Pitch:** fine pitch.
- **Relative:** when enabled, pitch changes are relative to the current pitch.
- **Phase Reset:** restarts all operators and resets the waveform to its start.

## OPL (drums) instrument editor

---

this is similar to the OPL instrument editor, but sets the parameters of snare, tom, top and hi-hat directly once a drums instrument is activated.

# OPLL FM synthesis instrument editor

---

the OPLL FM editor is divided into 5 tabs:

- **FM**: for controlling the basic parameters of FM sound source.
- **Macros (FM)**: for macros controlling algorithm, waveform and feedback.
- **Macros (OP1)**: for macros controlling FM parameters of operator 1.
- **Macros (OP2)**: for macros controlling FM parameters of operator 2.
- **Macros**: for other macros (volume/arp/pitch/patch).

## FM

---

the OPLL synthesizer is two-operator, meaning it takes two oscillators to produce a single sound. however, unlike the other FM chips, only one custom patch may be set at a time, shared among all 9 channels.

but don't worry! there also are 15 preset patches that you may select at any time.

these apply to the instrument as a whole:

- **Feedback (FB)**: determines how many times operator 1 returns its output to itself (0 to 7).
- **Sustain (SUS)**: enables the sustain flag (sets the release rate to 5).
- **Octave**: sets the "block" of the frequency register, which affects note range and precision.
- **OP2 Half Sine (DC)**: sets the waveform produced by carrier operator to half-sine.
- **OP1 Half Sine (DM)**: sets the waveform produced by modulator operator to half-sine.
- **preset dropdown**: selects OPLL preset instrument.
- this is the selector for the preset patches I mentioned before.
- once a preset patch is selected, only the volume is configurable.
- only one user-specified patch may be applied at a time!

if you select the special Drums patch, you may use the instrument in Drums mode of OPLL. an extra setting also appears:

- **Fixed frequency mode**: allows you to set a fixed frequency for the drum channels.

these apply to each operator:

- the crossed-arrows button can be dragged to rearrange operators.
- **Amplitude Modulation (AM)**: makes the operator affected by LFO tremolo.
- **Envelope generator sustain flag (EGS)**: when enabled, value of Sustain Level is in effect.
- **Attack Rate (AR)**: determines the rising time for the sound. the bigger the value, the faster the attack (0 to 15).
- **Decay Rate (DR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. it's the initial amplitude decay rate (0 to 15).

- **Sustain Level (SL)**: determines the point at which the sound ceases to decay and changes to a sound having a constant level. the sustain level is expressed as a fraction of the maximum level (0 to 15).
- **Release Rate (RR)**: determines the rate at which the sound disappears after note off. the higher the value, the shorter the release (0 to 15).
- **Total Level (TL)**: represents the envelope's highest amplitude, with 0 being the largest and 63 (decimal) the smallest. a change of one unit is about 0.75 dB.
- in the case of the second operator, it goes from 0 to 15 instead.
- **Key Scale Level (KSL)**: also known as "Level Scale". determines the degree to which the amplitude decreases according to the pitch.



- **Envelope Scale (KSR)**: also known as "Key Scale". determines the degree to which the envelope execution speed increases according to the pitch.
- **Frequency Multiplier (MULT)**: sets the coarse pitch offset in relation to the note (0 to 15). the values follow the harmonic scale. for example, 0 is -1 octave, 1 is 0 octaves, 2 is 1 octave, 3 is 1 octave 7 semitones, and so on.
- note that values 11, 13 and 14 behave as 10, 12 and 12 respectively.
- **Vibrato (VIB)**: makes the operator affected by LFO vibrato.

## macros

---

these macros allow you to control several parameters of FM per tick.

## FM Macros

---

all parameters are listed above.

## OP1-OP4 Macros

---

all parameters are listed above.

# Macros

---

- **Arpeggio**: pitch change sequence.
- **Patch**: changes the playing preset mid-note.
- through use of this macro, you may unlock different glitched sounds. useful for distortion guitars!
- **Pitch**: fine pitch.
- **Relative**: when enabled, pitch changes are relative to the current pitch.
- **Phase Reset**: restarts all operators and resets the waveform to its start.

# links

---

[FM instrument tutorial](https://www.youtube.com/watch?v=wS8edjurjDw) (<https://www.youtube.com/watch?v=wS8edjurjDw>) : A great starting point to learn how create and work with FM sounds. this was made for DefleMask, but all the same principles apply.

# FM (OPM) instrument editor

---

the FM editor is divided into 7 tabs:

- **FM**: for controlling the basic parameters of FM sound source.
- **Macros (FM)**: for macros controlling algorithm, feedback and LFO.
- **Macros (OP1)**: for macros controlling FM parameters of operator 1.
- **Macros (OP2)**: for macros controlling FM parameters of operator 2.
- **Macros (OP3)**: for macros controlling FM parameters of operator 3.
- **Macros (OP4)**: for macros controlling FM parameters of operator 4.
- **Macros**: for other macros (volume/arp/pitch/noise).

## FM

---

OPM is four-operator, meaning it takes four oscillators to produce a single sound.

these apply to the instrument as a whole:

- **Feedback (FB)**: determines how many times operator 1 returns its output to itself (0 to 7).
- **Algorithm (ALG)**: determines how operators are connected to each other (0 to 7).
- left-click pops up a small "operators changes with volume?" dialog where each operator can be toggled to scale with volume level.
- right-click to switch to a preview display of the waveform generated on a new note:
  - left-click restarts the preview.
  - middle-click pauses and unpauses the preview.
  - right-click returns to algorithm view.
- **LFO > Freq (FMS)**: determines how much will LFO have an effect in frequency (0 to 7).
- **LFO > Amp (AMS)**: determines how much will LFO have an effect in volume (0 to 3).
- only applies to operators which have AM turned on.

these apply to each operator:

- the crossed-arrows button can be dragged to rearrange operators.
- the **OP1**, **OP2**, **OP3**, and **OP4** buttons enable or disable those operators.
- **Amplitude Modulation (AM)**: makes the operator volume affected by LFO.
- **Attack Rate (AR)**: determines the rising time for the sound. the bigger the value, the faster the attack (0 to 31).
- **Decay Rate (DR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. it's the initial amplitude decay rate (0 to 31).
- **Sustain Level (SL)**: determines the point at which the sound ceases to decay and changes to a sound having a constant level. the sustain level is expressed as a fraction of the maximum level (0 to 15).
- **Decay Rate 2 (D2R) / Sustain Rate (SR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. this is the long "tail" of the sound that continues as long as the key is depressed (0 to 31).
- **Release Rate (RR)**: determines the rate at which the sound disappears after note off. the higher the value, the shorter the release (0 to 15).
- **Total Level (TL)**: represents the envelope's highest amplitude, with 0 being the largest and 127 (decimal) the smallest. a change of one unit is about 0.75 dB.



- **Envelope Scale (RS/KS)**: also known as "Key Scale" or "Rate Scale". determines the degree to which the envelope execution speed increases according to the pitch (0 to 3).
- **Frequency Multiplier (MULT)**: sets the coarse pitch offset in relation to the note (0 to 15). the values follow the harmonic scale. for example, 0 is -1 octave, 1 is 0 octaves, 2 is 1 octave, 3 is 1 octave 7 semitones, and so on.
- **Fine Detune (DT)**: shifts the pitch a little (0 to 7).
- **Coarse Detune (DT2)**: shifts the pitch by tens of cents (0 to 3).

## macros

these macros allow you to control several parameters of FM per tick.

### FM Macros

- **Algorithm**,
- **Feedback**,
- **LFO > Freq**,
- **LFO > Amp**: as described above.
- **AM Depth**: amplitude modulation depth.
- **PM Depth**: pitch modulation depth.
- **LFO Speed**: LFO frequency.
- **LFO Shape**: LFO shape. choose between saw, square, triangle, and random.
- **OpMask**: toggles each operator.

### OP1-OP4 Macros

all parameters are listed above.

# Macros

---

- **Arpeggio**: pitch sequence.
- **Noise Frequency**: specifies the noise frequency.
- this only applies to operator 4 of channel 8!
- **Panning**: toggles output on left and right channels.
- **Pitch**: fine pitch.
- **Relative**: when enabled, pitch changes are relative to the current pitch.
- **Phase Reset**: restarts all operators and resets the waveform to its start.

# links

---

[FM instrument tutorial](https://www.youtube.com/watch?v=wS8edjurjDw) (<https://www.youtube.com/watch?v=wS8edjurjDw>) : A great starting point to learn how create and work with FM sounds. this was made for DefleMask, but all the same principles apply.

# FM (OPN) instrument editor

---

the FM editor is divided into 7 tabs:

- **FM**: for controlling the basic parameters of FM sound source.
- **Macros (FM)**: for macros controlling algorithm, feedback and LFO.
- **Macros (OP1)**: for macros controlling FM parameters of operator 1.
- **Macros (OP2)**: for macros controlling FM parameters of operator 2.
- **Macros (OP3)**: for macros controlling FM parameters of operator 3.
- **Macros (OP4)**: for macros controlling FM parameters of operator 4.
- **Macros**: for other macros (volume/arp/pitch).

## FM

---

OPN is four-operator, meaning it takes four oscillators to produce a single sound.

these apply to the instrument as a whole:

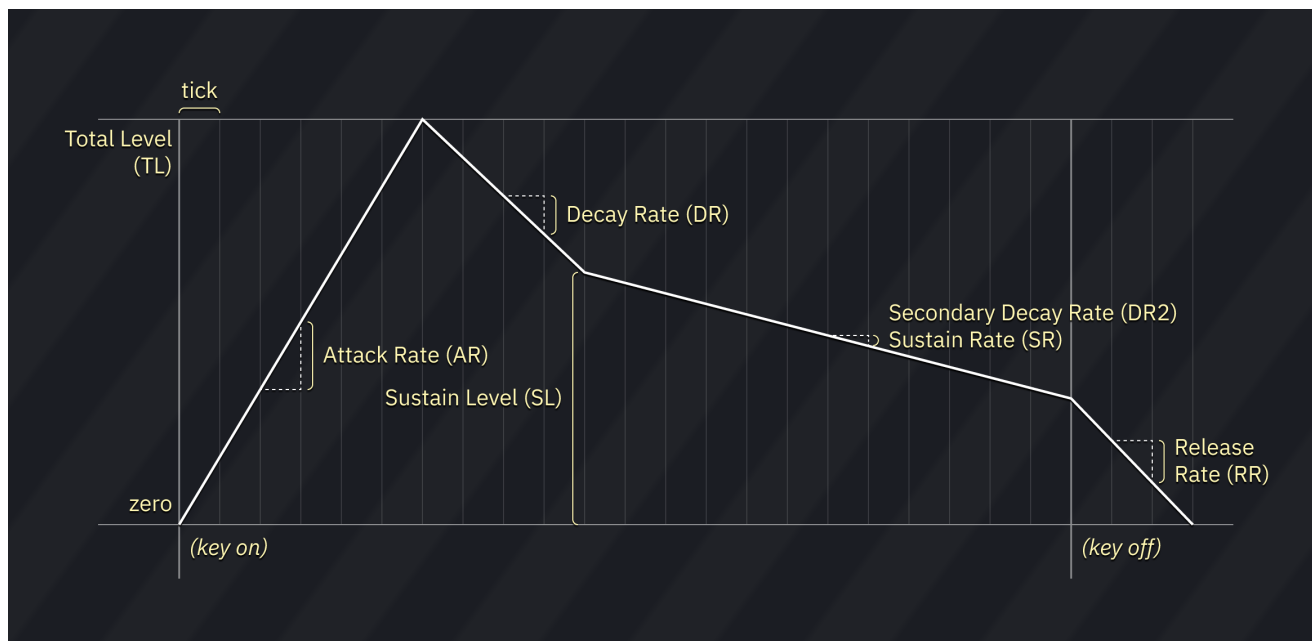
- **Algorithm (ALG)**: determines how operators are connected to each other (0 to 7).
- left-click pops up a small "operators changes with volume?" dialog where each operator can be toggled to scale with volume level.
- right-click to switch to a preview display of the waveform generated on a new note:
  - left-click restarts the preview.
  - middle-click pauses and unpauses the preview.
  - right-click returns to algorithm view.
- **Feedback (FB)**: determines how many times operator 1 returns its output to itself (0 to 7).
- **LFO > Freq (FMS)**: determines how much will LFO have an effect in frequency (0 to 7).
- **LFO > Amp (AMS)**: determines how much will LFO have an effect in volume (0 to 3).
- only applies to operators which have AM turned on.
- does not apply to YM2203.
- **Octave**: sets the "block" of the frequency register, which affects note range and precision.

these apply to each operator:

- the crossed-arrows button can be dragged to rearrange operators.
- the **OP1**, **OP2**, **OP3**, and **OP4** buttons enable or disable those operators.
- **Amplitude Modulation (AM)**: makes the operator's volume affected by LFO.
- does not apply to YM2203.
- **Attack Rate (AR)**: determines the rising time for the sound. the bigger the value, the faster the attack (0 to 31).
- **Decay Rate (DR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. it's the initial amplitude decay rate (0 to 31).
- **Sustain Level (SL)**: determines the point at which the sound ceases to decay and changes to a sound having a constant level. the sustain level is expressed as a fraction of the maximum level (0 to 15).
- **Decay Rate 2 (D2R) / Sustain Rate (SR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. this is the long "tail" of the sound that continues as long as the key is depressed (0 to 31).



- **Release Rate (RR)**: determines the rate at which the sound disappears after note off. the higher the value, the shorter the release (0 to 15).
- **Total Level (TL)**: represents the envelope's highest amplitude, with 0 being the largest and 127 (decimal) the smallest. a change of one unit is about 0.75 dB.
- **Hardware Envelope Generator (SSG-EG)**: executes the built-in envelope, inherited from AY-3-8910 PSG. speed of execution is controlled via envelope parameters.



- **Envelope Scale (RS/KS)**: also known as "Key Scale" or "Rate Scale". determines the degree to which the envelope execution speed increases according to the pitch (0 to 3).
- **Frequency Multiplier (MULT)**: sets the coarse pitch offset in relation to the note (0 to 15). the values follow the harmonic scale. for example, 0 is -1 octave, 1 is 0 octaves, 2 is 1 octave, 3 is 1 octave 7 semitones, and so on.
- **Fine Detune (DT)**: shifts the pitch a little (0 to 7).

## macros

these macros allow you to control several parameters of FM per tick.

### FM Macros

- **Algorithm**,
- **Feedback**,
- **LFO > Freq**,
- **LFO > Amp**: as described above.
- **LFO Speed**: LFO frequency.
- **OpMask**: toggles each operator.

### OP1-OP4 Macros

all parameters are listed above.

# Macros

---

- **Arpeggio**: pitch change sequence in semitones.
- **Panning**: toggles output on left and right channels.
- **Pitch**: fine pitch.
- **Relative**: when enabled, pitch changes are relative to the current pitch.
- **Phase Reset**: restarts all operators and resets the waveform to its start.

# links

---

[FM instrument tutorial](https://www.youtube.com/watch?v=wS8edjurjDw) (<https://www.youtube.com/watch?v=wS8edjurjDw>) : A great starting point to learn how create and work with FM sounds. this was made for DefleMask, but all the same principles apply.

# FM (OPZ) instrument editor

---

the FM editor is divided into 7 tabs:

- **FM**: for controlling the basic parameters of FM sound source.
- **Macros (FM)**: for macros controlling algorithm, feedback and LFO
- **Macros (OP1)**: for macros controlling FM parameters of operator 1
- **Macros (OP2)**: for macros controlling FM parameters of operator 2
- **Macros (OP3)**: for macros controlling FM parameters of operator 3
- **Macros (OP4)**: for macros controlling FM parameters of operator 4
- **Macros**: for other macros (volume/arp/pitch/noise).

## FM

---

OPZ is four-operator, meaning it takes four oscillators to produce a single sound.

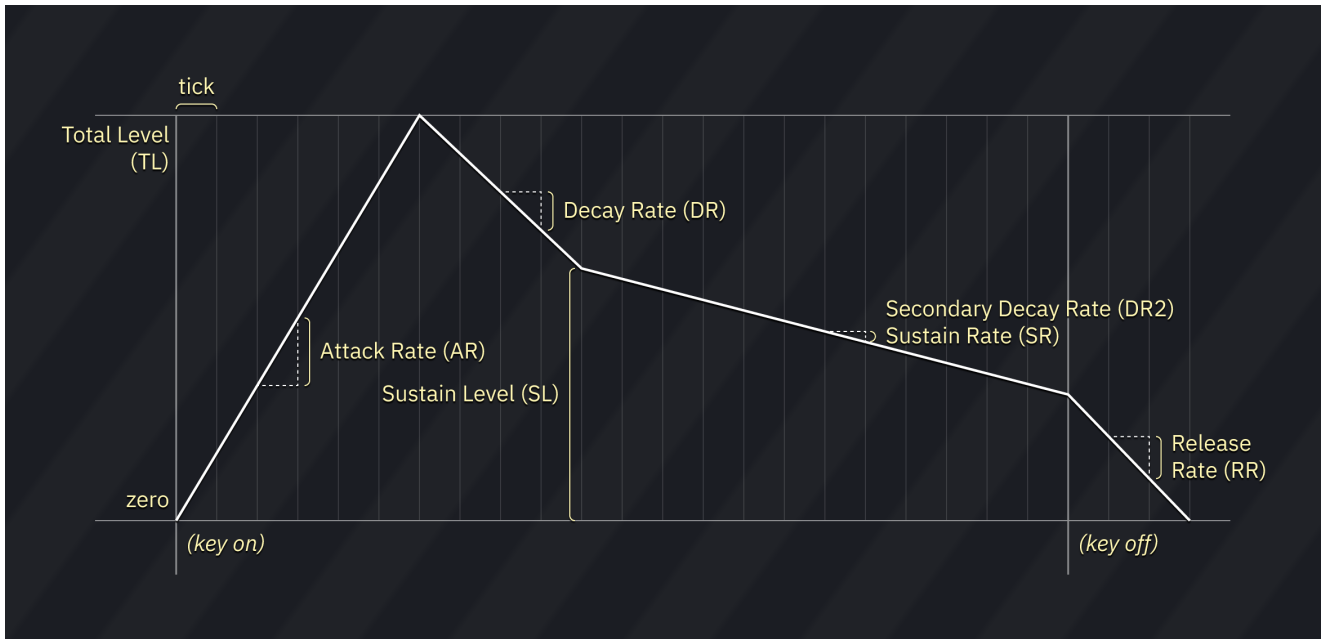
these apply to the instrument as a whole:

- **Algorithm (ALG)**: determines how operators are connected to each other (0 to 7).
- left-click pops up a small "operators changes with volume?" dialog where each operator can be toggled to scale with volume level.
- right-click to switch to a preview display of the waveform generated on a new note:
  - left-click restarts the preview.
  - middle-click pauses and unpauses the preview.
  - right-click returns to algorithm view.
- **Feedback (FB)**: determines how many times operator 1 returns its output to itself (0 to 7).
- **LFO > Freq (FMS/PMS)**: determines how much will LFO have an effect in frequency (0 to 7).
- **LFO > Amp (AM)**: determines how much will LFO have an effect in volume (0 to 3).
- **LF02 > Freq (FMS/PMS2)**: determines how much will the second LFO have an effect in frequency (0 to 7).
- **LF02 > Amp (AMS2)**: determines how much will the second LFO have an effect in volume (0 to 3).
- **Request from TX81Z**: if a Yamaha TX81Z is plugged in as MIDI input and output device, this sends a SysEx to the device in order to fetch its current voice.

these apply to each operator:

- the crossed-arrows button can be dragged to rearrange operators.
- **Amplitude Modulation (AM)**: makes the operator's volume affected by LFO.
- **Attack Rate (AR)**: determines the rising time for the sound. the bigger the value, the faster the attack (0 to 31).
- **Decay Rate (DR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. it's the initial amplitude decay rate (0 to 31).
- **Sustain Level (SL)**: determines the point at which the sound ceases to decay and changes to a sound having a constant level. the sustain level is expressed as a fraction of the maximum level (0 to 15).
- **Decay Rate 2 (D2R) / Sustain Rate (SR)**: determines the diminishing time for the sound. the higher the value, the shorter the decay. this is the long "tail" of the sound that continues as long as the key is depressed (0 to 31).

- **Release Rate (RR)**: determines the rate at which the sound disappears after note off. the higher the value, the shorter the release (0 to 15).
- **Total Level (TL)**: represents the envelope's highest amplitude, with 0 being the largest and 127 (decimal) the smallest. a change of one unit is about 0.75 dB.



- **Envelope Scale (RS/KS)**: also known as "Rate Scale" or "Key Scale". determines the degree to which the envelope execution speed increases according to the pitch (0 to 3).
- **Frequency Multiplier (MULT)**: sets the coarse pitch offset in relation to the note (0 to 15). the values follow the harmonic scale. for example, 0 is -1 octave, 1 is 0 octaves, 2 is 1 octave, 3 is 1 octave 7 semitones, and so on.
- **Fine Frequency Multiplier (Fine)**: a fine control for MULT.
- **Envelope Generator Shift (EGS)**: adds a "handicap" to the envelope. in other words, the minimum volume of the operator.
  - 0: no change
  - 1: -12dB
  - 2: -24dB
  - 3: -48dB
  - does not apply for OP4.
- **Reverb (REV)**: not a true reverb. extends release time, giving a slight reverb-like effect to the operator.
- **Fine Detune (DT)**: shifts the pitch a little (0 to 7).
- **Waveform Select (WS)**: changes the waveform of the operator.
- **Coarse Detune (DT2)**: shifts the pitch by tens of cents (0 to 3).

I am familiar with Yamaha TX81Z. where's LS and KVS?

these are software effects.

- you may use TL effects to simulate LS.
- you may access a KVS-like feature by clicking on the algorithm preview.

## fixed frequency mode

each operator has a Fixed Frequency mode. once enabled, the operator runs at the specified frequency regardless of the note.

## macros

---

these macros allow you to control several parameters of FM per tick.

## FM Macros

---

- **Algorithm**,
- Feedback**,
- LFO > Freq**,
- LFO > Amp**,
- LF02 > Freq**,
- LF02 > Amp**: as described above.
- **AM Depth**: amplitude modulation depth.
- **PM Depth**: pitch modulation depth.
- **LFO Speed**: LFO frequency.
- **LFO Shape**: LFO shape. choose between saw, square, triangle, and random.
- **AM Depth 2**: amplitude modulation depth (second LFO).
- **PM Depth 2**: pitch modulation depth (second LFO).
- **LF02 Speed**: LFO 2 frequency.
- **LF02 Shape**: LFO 2 shape. choose between saw, square, triangle, and random.

## OP1-OP4 Macros

---

most parameters are listed above.

## Macros

---

- **Arpeggio**: pitch change sequence in semitones.
- **Noise Frequency**: specifies the noise frequency.
- this only applies to operator 4 of channel 8!
- **Panning**: toggles output on left and right channels.
- **Pitch**: fine pitch.
- **Relative**: when enabled, pitch changes are relative to the current pitch.
- **Phase Reset**: restarts all operators and resets the waveform to its start. effectively the same as a 0Cxx retrigger.

## links

---

**FM instrument tutorial** (<https://www.youtube.com/watch?v=wS8edjurjDw>) : A great starting point to learn how create and work with FM sounds. this was made for DefleMask, but all the same principles apply.

# Irem GA20 instrument editor

---

the GA20 instrument editor contains three tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of sample.

# Game Boy instrument editor

---

the Game Boy instrument editor consists of three tabs: one controlling envelope of sound channels, another for the wave synth and macro tab containing several macros.

## Game Boy

---

- **Use software envelope:** switch to volume macro instead of envelope.
- this exploits a bug in the Game Boy sound hardware in order to have software envelopes ("zombie mode").
- only a couple emulators have accurate reproduction of this bug.
- **Initialize envelope on every note:** forces a volume reset on each new note.
- **Volume:** initial channel volume (0 to 15).
- **Length:** envelope decay/attack duration (0 to 7)
- **Sound Length:** cuts off channel after specified length, overriding the Length value.
- **Direction:** up makes the envelope an attack. down makes it decay.
- note: for attack to have effect, start at a lower volume.
- **Hardware Sequence:** this allows you to define a sequence of hardware envelope changes for creating complex envelopes. see the next section for more information.

## hardware sequence

Furnace provides a sequencer for the hardware envelope. this way you can define timed envelope changes which may be used for simulating ADSR, adding simple release, and other things.

the sequence consists of a list of "commands".

the + button adds a new command, which may be one of the following:

- **Envelope:** sets envelope values and retriggers note. it is highly recommended to have this as the first command.
- **Sweep:** sets sweep parameters. only works on the first channel.
- **Wait:** waits a specific number of ticks.
- **Wait for Release:** waits until the note is released with === or REL.
- **Loop:** goes to a previous position in the sequence.
- **Loop until Release:** same as Loop, but doesn't have effect after releasing the note.

each command in the sequence is represented in three columns:

- **Tick:** the tick this command will execute, followed by position in the sequence.
- **Command:** the command and its parameters.
- **Move/Remove:** allows you to move the command, or remove it.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

notes:

- only for Wave channel.
- on Game Boy, using the wave synth may result in clicking and/or phase resets. by default Furnace attempts to mitigate this problem though, but some clicking may still be audible.

## Macros

---

- **Volume:** volume sequence.
- note: this only appears if "Use software envelope" is checked.
- **Arpeggio:** pitch sequence.
- **Duty/Noise:** pulse wave duty cycle or noise mode sequence.
- **Waveform:** channel 3 wavetable sequence.
- **Panning:** output for left and right channels.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.



# GBA DMA instrument editor

---

the GBA DMA instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Waveform:** waveform sequence.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.

# GBA MinMod instrument editor

---

the GBA MinMod instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Waveform:** waveform sequence.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.
- **Special:** invert outputs.

# Konami K007232 instrument editor

---

the K007232 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of sample.

# K053260 instrument editor

---

the K053260 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>163</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Panning:** stereo panning sequence.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of sample.

# Atari Lynx instrument editor

---

Atari Lynx instrument editor consists of two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>163</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

note that using samples on Lynx is CPU expensive!

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Duty/Int**: bit pattern for LFSR taps and integration.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform/LFSR reset.
- **Load LFSR**: load a value to the LFSR.

## audio generation description

---

Atari Lynx generates sound using a 12-bit linear feedback shift register (LFSR) with configurable tap. nine separate bits can be enabled to be the source of feedback: 0, 1, 2, 3, 4, 5, 7, 10 and 11. to generate *any* sound at least one bit *must* be enabled.

### LFSR-based synthesis

a linear-feedback shift register is one method used for random number generation. it works by shifting a sequence of binary numbers (bits), taking the last bit into the output. then some of the bits are combined with others, doing a XOR (exclusive or) operation and then being pushed back.

think of it as a conveyor carrying glass bottles. each bottle may be empty or carrying water. the bottle at the end is taken. if there's water, then the output is 1. if it's empty, the output is 0. depending on the LFSR configuration, many bottles at specific positions ("taps") are looked at. these are combined from left to right, two by two:

- if two bottles are identical, an empty bottle is pushed.
  - if one bottle has water but the other is empty, a water bottle is pushed.
- the process is repeated indefinitely.

unlike PowerNoise, Lynx's taps are in fixed positions, but it has many of them.

## **square wave**

the LFSR is shifted at the rate define by sound pitch and generates square wave by setting channel output value to +volume or -volume, depending on the bit shifted in.

## **triangle wave**

alternatively when "int" bit is set sound wave is generated by adding or subtracting volume from output effectively producing triangle wave.

### **how triangle wave works?**

hint: to obtain triangle set bits "int" and "11" in "Duty/Int" sequence and set volume to about 22. by enabling 11th tap bit the value shifted in is negated after 11 bit is shifted in hence the volume is added for 11 cycles and then subtracted for 11 cycles.

# MSM5232 instrument editor

---

the instrument editor for MSM5232 consists of these macros:

- **Volume**: volume sequence.
- only has effect when the envelope mode of a group is set to External.
- **Arpeggio**: pitch sequence.
- **Group Ctrl**: group control sequence:
- **sustain**: enable sustain mode.
- **2'**: enable 2' overtone.
- **4'**: enable 4' overtone.
- **8'**: enable 8' overtone.
- **16'**: enable 16' overtone.
- **Group Attack**: set attack rate of group.
- **Group Decay**: set decay rate of group.
- **Noise**: toggle noise mode.

# MSM6258 instrument editor

---

the MSM6258 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>163</sup>.

note that samples on MSM6258 are tied to one frequency.

## Macros

---

- **Frequency Divider:** selects the frequency divider for the output.
- 0: divide by 512.
- 1: divide by 768.
- 2: divide by 1024.
- **Panning:** toggle left/right output.
- **Phase Reset:** trigger restart of sample.
- **Clock Divider:** clock divider sequence. when it is 1, the clock is divided by half.



# MSM6295 instrument editor

---

the MSM6295 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>163</sup>.

note that samples on MSM6295 are tied to one frequency.

## Macros

---

- **Volume:** volume sequence.
- **Frequency Divider:** selects the frequency divider for the output.
  - 0: divide by 132.
  - 1: divide by 165.
- **Phase Reset:** trigger restart of sample.

# MultiPCM instrument editor

the MultiPCM instrument editor contains three tabs: Sample, MultiPCM and Macros.

## Sample

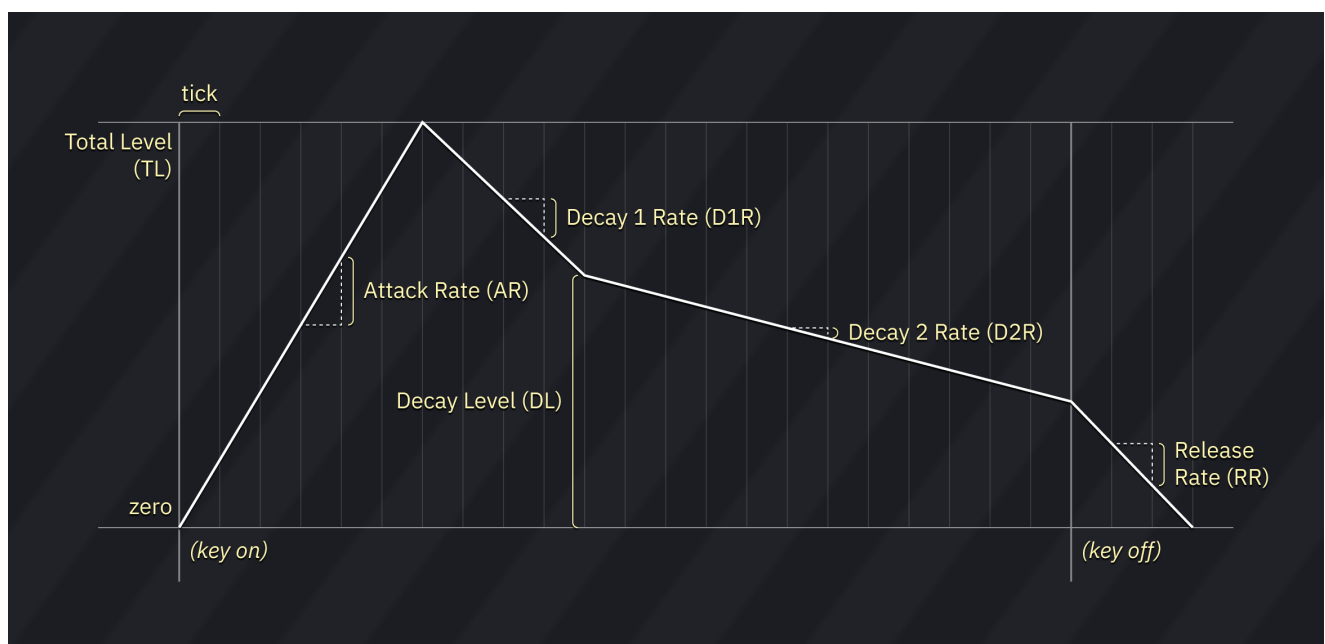
for sample settings, see [the Sample instrument editor](#)<sup>163</sup>.

## MultiPCM

MultiPCM contains an ADSR envelope, not unlike Yamaha OPN/OPM envelopes, and simple LFO, also similar to that found in OPN.

you may use this tab to set up MultiPCM-specific parameters:

- **AR**: sets Attack Rate.
- **D1R**: sets Decay 1 Rate.
- **DL**: sets Decay Level (analogue of Sustain Level on OPN chips).
- **D2R**: sets Decay 2 Rate (a.k.a. SR, Sustain Rate).
- **RR**: sets Release Rate.



- **RC**: sets Rate Correction amount. similar to Key Scale or Rate Scale in FM parameters, this determines the degree to which the envelope execution speed increases according to the pitch.
- **LFO Rate**: sets speed of LFO.
- **PM Depth**: sets depth of LFO vibrato.
- **AM Depth**: sets depth of LFO tremolo/amplitude modulation.
- **Damp**: enforce quickly fading out the sample over 11 ms.
- **Pseudo Reverb**: enables reverb-like effect.
- **LFO Reset**: disables and resets LFO.
- **Disable volume change ramp**: if set to "on", volume change interpolation is disabled.

# Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Panning:** output level for left and right channels (from -3 to +3).
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.
- **LFO Speed:** sequence of LFO speeds.
- **LFO Vib depth:** LFO vibrato sequence.
- **LFO Vib depth:** LFO tremolo sequence.

# Namco 163 instrument editor

---

the Namco 163 instrument editor consists of three tabs: "Namco 163" for control of various waveform parameters, "Wavetable" for control of the wave synth and "Macro" containing several macros.

## Namco 163

---

- **Load waveform:** if enabled, a waveform will be loaded when this instrument plays.
- if it isn't then only the position/length change.
- **Waveform:** determines the waveform that will be loaded.
- only appears when Load waveform is enabled.
- **Per-channel wave position/length:** when enabled, the position/length settings are split per channel.
- **Position:** determines the waveform position in RAM.
- **Length:** determines the waveform length in RAM.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

note that setting the Update Rate to something greater than 1 and playing the instrument in two channels may result in conflicts.

## Macros

---

- **Volume:** volume levels sequence.
- **Arpeggio:** pitch sequence.
- **Wave Pos:** sets waveform seek position in RAM.
- **Waveform:** sets waveform.
- **Pitch:** fine pitch.
- **Wave Length:** sets waveform length.

# Nintendo DS instrument editor

---

the Nintendo DS instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Duty:** pulse width sequence.
- only in PSG channels.
- **Panning:** left/right balance sequence.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.

# NES instrument editor

---

the NES instrument editor consists of two tabs.

## DPCM

---

this tab is somewhat similar to [the Sample instrument editor](#)<sup>183</sup>, but it has been tailored for use with NES' DPCM channel.

- **Sample:** specifies which sample should be assigned to the instrument.
- **Use sample map:** enables mapping different samples to notes. see next section for more information.
- when this option is disabled, 16 notes (from C-0 to D#1 and repeating) will map to the DPCM channel's 16 possible pitches.

### sample map

the sample map allows you to set a sample for each note.

after enabling this option, a table appears with the contents of the sample map.

- the first column represents the input note.
- the second column allows you to type in a sample number for each note.
- you may press Delete to clear it.
- the third one is used to set the DPCM pitch at which the specified sample will play.
- for possible values, refer to the table below.
- you may press Delete to clear it. if no value is specified, the last pitch is used.
- the fourth column allows you to set the initial delta counter value when playing the sample.
- this is an hexadecimal number.
- the range is 00 to 7F.
- you may press Delete to clear it. if no value is specified, the delta counter isn't altered.
- the fifth and last column provides a combo box for selecting a sample.

you may right-click anywhere in the number, pitch and delta columns for additional options:

- **set entire map to this pitch:** sets the DPCM pitch of all notes to the selected cell's.
- **set entire map to this delta counter value:** sets the initial delta counter value of all notes to the selected cell's.
- **set entire map to this sample:** sets the sample number of all notes to the selected cell's.
- **reset pitches:** resets the sample map's DPCM pitches to defaults (15).
- **clear delta counter values:** removes all delta counter values from the map.
- **clear map samples:** removes all samples from the map.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Duty/Noise:** duty cycle and noise mode.

- pulse duty cycles:
  - 0: 12.5%
  - 1: 25%
  - 2: 50%
  - 3: 75%
- noise modes:
  - 0: long noise
  - 1: short noise
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform.

# NEC PC Engine instrument editor

---

the PCE instrument editor contains three tabs: Sample, Wavetable and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

note: on PC Engine, using the wave synth may result in clicking and/or phase resets. by default Furnace attempts to mitigate this problem though.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Noise**: enable noise mode.
- only on channels 5 and 6.
- **Waveform**: wavetable sequence.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform.



# Commodore PET instrument editor

---

the PET instrument editor consists of these macros:

- **Volume:** volume sequence (on/off).
- **Arpeggio:** pitch sequence.
- **Waveform:** an 8/1 waveform.
- **Pitch:** fine pitch sequence.

# Pokémon Mini/QuadTone instrument editor

---

used in these two chips/systems. these macros are available:

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Pulse Width:** pulse width sequence.
- **Pitch:** fine pitch.

# Atari POKEY instrument editor

---

the instrument editor for POKEY consists of these macros:

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **AUDCTL**: audio control register sequence:
- **poly9**: reduces size of LFSR. only on noise waveforms!
- **high1**: runs channel 1 at 1.79MHz.
- **high3**: runs channel 3 at 1.79MHz.
- **16-bit 1+2**: enables 16-bit frequency mode by combining channels 1 and 2.
- **16-bit 3+4**: enables 16-bit frequency mode by combining channels 3 and 4.
- **filter 1+3**: applies a high-pass "filter" by combining channels 1 and 3.
- **filter 2+4**: applies a high-pass "filter" by combining channels 2 and 4.
- **15KHz**: runs channels at 15KHz.
- **Waveform**: wave selection sequence:
  - 0: harsh noise (poly5+17)
  - 1: square buzz (poly5)
  - 2: weird noise (poly4+5)
  - 3: square buzz (poly5)
  - 4: soft noise (poly17)
  - 5: square
  - 6: bass (poly4)
  - 7: buzz (poly4)
- **Pitch**: fine pitch.

# PowerNoise instrument editor

---

the PowerNoise instrument editor consists of two tabs.

## LFSR-based synthesis

---

PowerNoise employs LFSR-based synthesis for the noise channels, using linear-feedback shift registers for sound generation.

a linear-feedback shift register is one method used for random number generation. it works by shifting a sequence of binary numbers (bits), taking the last bit into the output. then one of the bits is either pushed back into the register, or combined with another, doing a XOR (exclusive or) operation and then being pushed back.

think of it as a conveyor carrying glass bottles. each bottle may be empty or carrying water. the bottle at the end is taken. if there's water, then the output is 1. if it's empty, the output is 0. depending on the LFSR configuration:

- a bottle is pushed into the conveyor. it is either empty or filled with water depending on the bottle at a specific position in the conveyor (this is called a "tap"), or
  - two bottles at specific positions ("taps") are looked at and combined as follows:
  - if the bottles are identical, an empty bottle is pushed.
  - if one bottle has water but the other is empty, a water bottle is pushed.
- the process is repeated indefinitely.

PowerNoise uses either one or two taps for the LFSR, configurable via the Control macro.

the LFSR must be initialized before it can produce sound. the Load LFSR macro allows you to do so.

by default the LFSR is configured to produce square waves, by having a single tap in position 1 and an alternating LFSR pattern.

## Macros (noise)

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger reloading the LFSR.
- **Control**: channel settings:
- **slope AM**: when enabled, this channel's output and the slope channel go through amplitude modulation. the final output plays in the slope channel.
- **tap B**: enables use of two taps for the LFSR.
- **Tap A Location**: sets the position of the first tap.

- **Tap B Location:** sets the position of the second tap.
- **Load LFSR:** allows you to load the LFSR with a specific pattern.

## PowerNoise tab

---

this tab allows you to change the base octave - important when you have set a longer LFSR pattern.

## PowerNoise (slope) instrument editor

---

this channel has its own instrument type, as it does not use LFSR-based synthesis but instead generates saw waves.

it uses a custom algorithm which will be (roughly) described below.

the slope channel uses two "portions" - each with length, offset, invert and clip parameters. the channel alternates between these portions as it is cycled.

on every cycle, the offset of the current portion is either added or subtracted into the accumulator (depending on the invert parameter), effectively behaving like a multiplier.

if the clip parameter is enabled, this will make sure the accumulator doesn't go past 0 or 127 (depending on the invert parameter, again). otherwise, the accumulator will be ANDed with 127.

once an amount of cycles set by the portion length parameter have elapsed, the channel switches into the other portion.

the current value of the accumulator is output.

## Macros (slope)

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Control:** channel settings:
- **clip A:** sets clip parameter of first portion.
- **clip B:** sets clip parameter of second portion.
- **reset A:** resets the first portion.
- **reset B:** resets the second portion.
- **invert A:** sets invert parameter of first portion.
- **invert B:** sets invert parameter of second portion.
- **Portion A Length:** sets the duration of the first portion.
- **Portion B Length:** sets the duration of the second portion.
- **Portion A Offset:** sets the accumulator speed of the first portion.
- **Portion B Offset:** sets the accumulator speed of the second portion.

# Sega PSG instrument editor

---

the instrument editor for Sega PSG, SMS, and other TI SN76489 derivatives consists of these macros:

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Duty:** noise mode.
- 0: short noise; preset frequencies.
- 1: long noise; preset frequencies.
- 2: short noise; use channel 3 for frequency.
- 3: long noise; use channel 3 for frequency.
- **Panning:** output for left and right channels.
- only on Game Gear!
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of **noise only**.

# PV-1000 instrument editor

---

the instrument editor for the Casio PV-1000 consists of these macros:

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Pitch:** fine pitch.

# Capcom QSound instrument editor

---

the QSound instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Echo Level:** echo wet volume sequence.
- **Panning:** stereo panning sequence.
- **Surround:** toggles whether QSound algorithm is enabled.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of sample.
- **Echo Feedback:** echo feedback sequence.
- note: global!
- **Echo Length:** echo length sequence.
- note: global!



# Ricoh RF5C68 instrument editor

---

the RF5C68 instrument editor contains two tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of sample.

# Philips SAA1099 instrument editor

---

the SAA1099 instrument editor consists of these macros:

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Duty/Noise**: noise generator frequency. the following values are available:
  - 0: high
  - 1: mid
  - 2: low
  - 3: use frequency of channel 1 or 4 (depending on where the instrument plays).
- **Waveform**: selector between tone and noise.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Envelope**: envelope generator settings:
- **enable**: enables the envelope generator.
- **N/A**: has no effect.
- **fixed**: toggles whether to use a fixed, slow frequency or lock to the frequency of channel 2 or 5 (depending on where the instrument plays).
- **resolution**: increases the envelope generator pitch resolution.
- **direction**: inverts the envelope.
- **cut**: cuts the envelope (producing saw wave out of tri wave)
- **loop**: toggles whether envelope is one-shot or constantly looping.
- **mirror**: sets whether the right output will mirror the left one.
- the envelope only has effect in channels 3 and 6.

# Amiga/PCM sound source instrument editor

---

the Generic Sample instrument editor consists of a sample selector and several macros:

## Sample

---

- **Sample:** specifies which sample should be assigned to the instrument.
- **Use wavetable:** uses wavetable instead of a sample.
- only available in Amiga and Generic PCM DAC.
- **Use sample map:** enables mapping different samples to notes. see next section for more in formation.

## sample map

the sample map allows you to set a sample for each note. this can be used to create more realistic instruments, split key instruments, drum kits and more.

after enabling this option, a table appears with the contents of the sample map.

- the first column represents the input note.
- the second column allows you to type in a sample number for each note.
- you may press Delete to clear it.
- the third one is used to set the note at which the specified sample will play.
- the fourth and last column provides a combo box for selecting a sample.

you may right-click anywhere in the number and note columns for additional options:

- **set entire map to this note:** sets the note number of all notes to the selected cell's.
- **set entire map to this sample:** sets the sample number of all notes to the selected cell's.
- **reset notes:** resets the sample map's notes to defaults (a chromatic scale).
- **clear map samples:** removes all samples from the map.

## Macros

---

- **Volume:** volume sequence. does not apply to some chips.
- **Arpeggio:** pitch sequence.
- **Waveform:** waveform sequence.
- only appears when "Use wavetable" is enabled.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.

# Konami SCC/Bubble System WSG instrument editor

---

the SCC/Bubble System WSG instrument editor consists of two tabs.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

be noted that channel 4 and 5 share the same waveform on SCC (non-plus). careful.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Waveform:** specifies wavetable.
- **Pitch:** fine pitch.

# SegaPCM instrument editor

---

the SegaPCM instrument editor contains three tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of sample.

# SID2 instrument editor

---

the SID2 instrument editor consists of two tabs: "SID2" to control various parameters of sound channels, and "Macros" containing several macros.

## SID2

---

- **Waveform:** allows selecting a waveform.
- more than one waveform can be selected at once. in that case a logical AND mix of waves will occur...
  - although with default mix mode it does occur a bit wrong (like on 8580 SID chip). see below what happens when other modes are in use.
- noise is an exception. it cannot be used with any of the other waveforms.
  - again, only when default mix mode is on.
- **Attack:** determines the rising time for the sound. the bigger the value, the slower the attack. (0 to 15).
- **Decay:** determines the diminishing time for the sound. the higher the value, the longer the decay (0 to 15).
- **Sustain:** sets the volume level at which the sound stops decaying and holds steady (0 to 15).
- **Release:** determines the rate at which the sound fades out after note off. the higher the value, the longer the release (0 to 15).
- **Duty:** specifies the width of a pulse wave (0 to 4095).
- **Reset duty on new note:** overwrite current duty value with the one that is specified in the instrument on new note.
  - only useful when using relative duty macro.
- **Ring Modulation:** when enabled, the channel's output will be multiplied with the previous channel's.
- **Oscillator Sync:** enables oscillator hard sync. as the previous channel's oscillator finishes a cycle, it resets the period of the channel's oscillator, forcing the latter to have the same base frequency. this can produce a harmonically rich sound, the timbre of which can be altered by varying the synchronized oscillator's frequency.
- **Enable filter:** when enabled, this instrument will go through the filter.
- **Initialize filter:** initializes the filter with the specified parameters:
- **Cutoff:** the filter's point in where frequencies are cut off (0 to 4095).
- **Resonance:** amplifies or focuses on the cutoff frequency, creating a secondary peak forms and colors the original pitch (0 to 255).
- **Filter mode:** sets the filter mode. you may pick one or more of the following:
  - **low:** a low-pass filter. the lower the cutoff, the darker the sound.
  - **high:** a high-pass filter. higher cutoff values result in a less "bassy" sound.
  - **band:** a band-pass filter. cutoff determines which part of the sound is heard (from bass to treble).
  - multiple filter modes can be selected simultaneously. for example, selecting both "low" and "high" results in a bandstop (notch) filter.
- **Noise Mode:** dictates how noise behaves.
  - 0 means usual "white" noise.
  - 1-3 provide different tonal waves (in other words, small excerpts of noise are looped, creating a wave with tonal sound). mode 1 provides more "pure" tonal sound while modes 2 and 3 provide harsh, rich sounds, which can be further modified by filtering them.

- when *only* noise wave is enabled, frequency calculation is altered in a way that this noise wave stays in tune, so wave can freely be used to play actual music.
- **Wave Mix Mode:** dictates how different waves on the same channel are mixed together.
- mode 0 does it the same way as on 8580 SID chip.
- mode 1 does a bitwise AND between all the enabled waves (including noise!).
- modes 2 and 3 operate in the same way, but they do bitwise OR and bitwise XOR.
- **Absolute Cutoff Macro:** when enabled, the cutoff macro will go from 0 to 4095, and it will be absolute (in other words, control the cutoff directly rather than being relative).
- **Absolute Duty Macro:** when enabled, the duty macro will go from 0 to 4095.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Pitch:** fine pitch.
- **Duty:** pulse width sequence.
- **Waveform:** select the waveform used by instrument.
- **Phase Reset:** trigger restart of envelope.
- **Cutoff:** filter cutoff.
- **Filter Mode:** select the filter mode.
- **Filter Toggle:** turns filter on and off.
- **Resonance:** filter resonance sequence.
- **Special:** ring and oscillator sync selector, as well as:
- **gate bit:**
  - set (1): key on. if previous state was 0 it triggers envelope start/restart; if previous state was 1, it does nothing.
  - reset (0): key off. if previous state was 1 it triggers envelope release; if previous state was 0, it does nothing.
- **Attack:** sets envelope attack speed.
- if you modify attack speed when the envelope is in attack phase it immediately changes.
- **Decay:** sets envelope decay speed.
- if you modify decay speed when envelope is in decay phase it immediately changes.
- **Sustain:** sets envelope sustain level.
- if you modify sustain level when envelope is in sustain phase it immediately changes. note that, unlike SID chips, you can change sustain level in any direction (both 2->5 and 5->2 work and do not trigger envelope release).
- **Release:** sets envelope release speed.
- if you modify release speed when envelope is in release phase it immediately changes.
- **Noise Mode:** select the noise mode.
- **Wave Mix:** select the waveform mix mode.

# SID3 instrument editor

---

the SID3 editor is divided into 8 tabs:

- **SID3**: for controlling the basic parameters of SID3 sound source.
- **Wavetable**: for controlling the wavetable synth.
- **Sample**: for various sample settings.
- **Macros (Filter 1)**: for macros controlling parameters of filter 1.
- **Macros (Filter 2)**: for macros controlling parameters of filter 2.
- **Macros (Filter 3)**: for macros controlling parameters of filter 3.
- **Macros (Filter 4)**: for macros controlling parameters of filter 4.
- **Macros**: for other macros.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

## SID3

---

- **Waveform**: allows selecting a waveform.
- more than one waveform can be selected at once. in that case a logical AND mix of waves will occur...
  - although with default mix mode it does occur a bit wrong (like on 8580 SID chip). see below what happens when other modes are in use.
- **Special wave**: allows selecting a special wave. the wave preview is to the right.
- **Wavetable channel**: replaces and hides some macros and UI elements, and makes instrument operate with last wavetable/sample channel:
- **Waveform** macro now selects a wavetable
- **Duty**, **Special Wave**, **Feedback**, **Noise Phase Reset**, **Noise LFSR bits** and **Wave Mix** macros are hidden
- **1-Bit Noise** macro now controls wavetable/PCM mode (it becomes **Sample Mode** macro)
- **Inv. left** and **Inv. right**: invert the signal of corresponding stereo channels.
- **Attack**: determines the rising time for the sound. the bigger the value, the slower the attack. (0 to 255).
- **Decay**: determines the diminishing time for the sound. the higher the value, the longer the decay (0 to 255).
- **Sustain**: sets the volume level at which the sound stops decaying and holds or also decays, but with different speed (0 to 255).



- **Sustain rate:** sets the speed at which the sound decays after reaching sustain volume level. (0 to 255).
- **Release:** determines the rate at which the sound fades out after note off. the higher the value, the longer the release (0 to 255).
- **Wave Mix Mode:** dictates how different waves on the same channel are mixed together.
- **Duty:** specifies the width of a pulse wave (0 to 65535).
- **Feedback:** specifies the feedback level (0 to 255).
- **Reset duty on new note:** overwrite current duty value with the one that is specified in the instrument on new note.
- only useful when using relative duty macro.
- **Absolute Duty Macro:** when enabled, the duty macro will go from 0 to 65535 (in other words, control the duty directly rather than being relative).
- **Ring Modulation:** when enabled, the channel's output will be multiplied with the source channel's.
- **Oscillator Sync:** enables oscillator hard sync. as the source channel's oscillator finishes a cycle, it resets the period of the channel's oscillator, forcing the latter to have the same base frequency. this can produce a harmonically rich sound, the timbre of which can be altered by varying the synchronized oscillator's frequency.
- **Phase Modulation:** when enabled, the channel's phase will be modified with the source channel's signal (signal is taken from filtered channel's output if filters are enabled).
- **Separate noise pitch:** when enabled, the noise frequency/pitch will be controllable via special macros: **Noise Arpeggio** and **Noise Pitch**.

Then follow controls for each of the 4 filters:

- **Enable filter:** when enabled, this instrument will go through the filter.
- **Initialize filter:** initializes the filter with the specified parameters:
- **Cutoff:** the filter's point in where frequencies are cut off (0 to 65535).
- **Resonance:** amplifies or focuses on the cutoff frequency, creating a secondary peak forms and colors the original pitch (0 to 255).
- **Filter mode:** sets the filter mode. you may pick one or more of the following:
  - **low:** a low-pass filter. the lower the cutoff, the darker the sound.
  - **high:** a high-pass filter. higher cutoff values result in a less "bassy" sound.
  - **band:** a band-pass filter. cutoff determines which part of the sound is heard (from bass to treble).
  - multiple filter modes can be selected simultaneously. for example, selecting both "low" and "high" results in a bandstop (notch) filter.
- **Output volume:** sets the filter output volume (0 to 255).
- **Distortion level:** dictates how hard the signal is distorted (soft clipping). distortion is slightly asymmetrical (0 to 255).
- **Absolute Cutoff Macro:** when enabled, the cutoff macro will go from 0 to 65535, and it will be absolute.
- **Change cutoff with pitch:** when enabled, the cutoff will be scaled according to the frequency offset from specified note.
- **Decrease cutoff when pitch increases:** if this is enabled, filter cutoff will decrease if you increase the pitch. if this is disabled, filter cutoff will increase if you increase the pitch.
- **Cutoff change center note:** this note marks the center frequency at which no cutoff scaling is happening. the further you go from it in each direction, the more the cutoff scaling will be.
- **Cutoff change strength:** how much cutoff will be scaled.
- **Change resonance with pitch:** when enabled, the resonance will be scaled according to the frequency offset from specified note.

- **Decrease resonance when pitch increases:** if this is enabled, filter resonance will decrease if you increase the pitch. if this is disabled, filter resonance will increase if you increase the pitch.
- **Resonance change center note:** this note marks the center frequency at which no resonance scaling is happening. the further you go from it in each direction, the more the resonance scaling will be.
- **Resonance change strength:** how much resonance will be scaled.
- **Filters connection matrix:** controls routing of the filters' signals.
- **In:** this column connects the filters to ADSR sound output.
- next 4 columns make up the inter-filters connection matrix.
- **Out:** this column connects the filters' output to final channel output.

## special noise LFSR configurations

this table contains a list of LFSR configurations that are automatically detected and brought to tune by Furnace. a short description is given. number needs to be pasted into **Noise LFSR bits** macro. it is recommended to place a single bar in **Noise Phase Reset** macro for the consistency of the wave.

LFSR CONFIG	DESCRIPTION
524288	wave very close to <a href="#">SID2<sub>186</sub></a> noise mode 1 wave. tonal, without very harsh overtones.
541065280	wave resembling vocals, has two main tones at least 2 octaves apart
2068	wave very close to SID2 noise mode 3 wave. tonal but with harsh timbre.
66	wave very close to SID2 noise mode 2 wave. timbre is somewhere in-between SID2's noise mode 1 and noise mode 3 waves.

if you find more interesting waves, please contact LTVA or tildearrow, so they can be added to Furnace frequency correction routine and to this table.

## Filter X macros

---

- **Cutoff:** filter x cutoff sequence.
- **Resonance:** filter x resonance sequence.
- **Filter toggle:** turns filter x on and off.
- **Filter mode:** select filter x mode.
- **Distortion level:** filter x distortion level sequence.
- **Output Volume:** filter x output volume sequence.
- **Channel Input Connection:** connect filter x to channel ADSR output.
- **Channel Output Connection:** connect filter x output to final channel output.
- **Connection Matrix Row:** connect other filters' outputs to filter x input.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Pitch:** fine pitch.
- **Duty:** pulse width sequence.
- **Waveform:** select the waveform used by instrument.
- in wavetable channel mode controls the wavetable index.

- **Special Wave:** select the special wave used by instrument.
- **Noise Arpeggio:** noise pitch sequence.
- this macro is visible only if **Separate noise pitch** option is enabled. otherwise noise pitch is controlled by **Arpeggio** and **Pitch** macros.
- **Noise Pitch:** fine pitch.
- this macro is visible only if **Separate noise pitch** option is enabled.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Channel Inversion:** invert signal of left and right channels.
- **Key On/Off:** envelope release/start again control.
- **Special:** ring, oscillator sync and phase modulation selector.
- **Ring Mod Source:** ring modulation source channel.
- **Hard Sync Source:** oscillator sync source channel.
- **Phase Mod Source:** phase modulation source channel.
- **Feedback:** feedback sequence
- **Phase Reset:** trigger restart of waveform.
- **Noise Phase Reset:** trigger restart of noise accumulator and LFSR.
- **Envelope Reset:** trigger restart of envelope (unlike key on/off, envelope is forced to restart from 0 volume level no matter which volume it is outputting now).
- **Attack:** sets envelope attack speed.
- if you modify attack speed when the envelope is in attack phase it immediately changes.
- **Decay:** sets envelope decay speed.
- if you modify decay speed when envelope is in decay phase it immediately changes.
- **Sustain:** sets envelope sustain level.
- if you modify sustain level when envelope is in sustain phase it immediately changes.
- **Sustain Rate:** sets envelope sustain rate.
- if you modify sustain rate when envelope is in sustain phase it immediately changes.
- **Release:** sets envelope release speed.
- if you modify release speed when envelope is in release phase it immediately changes.
- **Noise LFSR bits:** sets feedback bits of noise LFSR.
- **1-Bit Noise:** controls noise mode.
- in wavetable channel mode it's called **Sample Mode**, and macro controls wave/PCM mode of the last channel.
- **Wave Mix:** select the waveform mix mode.

# SM8521 instrument editor

---

the SM8521 instrument editor contains two tabs: Wavetable and Macros.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Waveform**: wavetable sequence.
- **Pitch**: fine pitch.

# SNES instrument editor

---

these four tabs are unique to the editor for SNES instruments.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## SNES

---

**Use envelope** enables the ADSR volume envelope. if it is on:

- **A**: attack rate.
- **D**: decay rate.
- **S**: sustain level.
- **D2**: decay rate during sustain.
- only appears when Sustain/release mode is Effective or Delayed.
- **R**: release rate.
- **Sustain/release mode**:
- **Direct**: note release acts as note cut.
- **Effective (linear decrease)**: after release, volume lowers by subtractions of 1/64 steps.
- **Effective (exponential decrease)**: after release, volume decays exponentially. see [gain chart](#)<sup>300</sup>.
- **Delayed (write R on release)**: after release, waits until A and D have completed before starting release.

if envelope is off:

- **Gain Mode**: selects gain mode.
- **Direct**: direct gain from 0 to 127.
- **Decrease (linear)**: linear gain from -0 to -31.
- **Decrease (logarithmic)**: exponential gain from -0 to -31.
  - note: using decrease modes will not produce any sound unless a Gain macro is set. the first tick must be the initial gain, and the second tick must be the decrease gain value. gain values are as described in the Macros section below.
- **Increase (linear)**: linear gain from +0 to +31.
- **Increase (bent line)**: inverse exponential gain from +0 to +31.
- **Gain**: value of gain.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

only active when Use wavetable is enabled in the Sample tab.

# Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Noise Freq**: frequency of noise generator.
- note: global!
- **Waveform**: waveform.
- only effective when Use wavetable is enabled.
- **Panning (left)**: output level of left channel.
- **Panning (right)**: output level of right channel.
- **Pitch**: fine pitch.
- **Special**: bitmap of flags.
- invert left: inverts output of left channel.
- invert right: inverts output of right channel.
- pitch mod: modulates pitch using previous channel's output.
- echo: enables echo.
- noise: enables noise generator.
- **Gain**: sets mode and value of gain.
- 0 to 127: direct gain from 0 to 127.
- 128 to 159: linear gain from -0 to -31 (decrease linear).
- 160 to 191: exponential gain from -0 to -31 (decrease exponential).
- 192 to 223: linear gain from +0 to +31 (increase linear).
- 224 to 255: exponential gain from +0 to +31 (increase bent line).

# tildearrow Sound Unit instrument editor

---

this instrument editor has two tabs.

## Sound Unit

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the differences are:

- the lack of an "Use wavetable" option
- the presence of a "Use sample" one
- the presence of a "**Switch roles of frequency and phase reset timer**" option. when enabled, this writes frequency to the phase reset timer register rather than the frequency register
- this may be used to create sync-like effects.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Duty/Noise**: waveform duty cycle sequence.
- **Waveform**: select waveform.
  - 0: pulse wave
  - 1: sawtooth
  - 2: sine wave
  - 3: triangle wave
  - 4: noise
  - 5: periodic noise
  - 6: XOR sine
  - 7: XOR triangle
- **Panning**: stereo panning sequence.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform.
- **Cutoff**: set filter cutoff.
- **Resonance**: set filter resonance.
- values that are too high may distort the output!
- **Control**: filter parameter/ring mod sequence.
- **band pass**: a band-pass filter. cutoff determines which part of the sound is heard (from bass to treble).
- **high pass**: a high-pass filter. higher cutoff values result in a less "bassy" sound.
- **low pass**: a low-pass filter. the lower the cutoff, the darker the sound.
- **ring mod**: enable ring modulation with previous channel.
  - note: square wave goes from 0 to volume, so in that case it acts more like amplitude modulation.

- **Phase Reset Timer:** sets the phase reset timer.
- if the "Switch roles of frequency and phase reset timer" option in the Sound Unit tab is enabled, this macro controls the frequency register instead.



# T6W28 instrument editor

---

the instrument editor for T6W28 consists of these macros:

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Noise Type**: noise type sequence:
  - 0: short noise
  - 1: long noise
- **Panning (left)**: output level for left channel.
- **Panning (right)**: output level for right channel.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform.

# TED instrument editor

---

TED instrument editor consists of these macros:

- **Volume:** volume sequence.
- note: global! affects entire chip.
- **Arpeggio:** pitch sequence.
- **Square/Noise:** select whether square, noise or nothing will be output.
- noise only available on channel 2
- if square and noise are enabled, square takes precedence.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.
- note: global! triggers both channels...

# Atari TIA instrument editor

---

the TIA instrument editor consists of these macros:

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- note: fixed mode works differently. it sets the frequency directly rather than the note, so it only goes from 0 to 31.
- **Waveform:** selects waveform to be used:
  - 0: nothing
  - 1: buzzy
  - 2: low buzzy
  - 3: flangy
  - 4: square
  - 5: square
  - 6: pure buzzy
  - 7: reedy
  - 8: noise
  - 9: reedy
  - A: pure buzzy
  - B: nothing
  - C: low square
  - D: low square
  - E: low pure buzzy
  - F: low reedy
- **Pitch:** "fine" pitch. fine in quotes as TIA doesn't have true pitch control at all.

# VERA instrument editor

---

VERA instrument editor consists of these macros:

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Duty:** pulse duty cycle sequence.
- **Waveform:** select the waveform used by instrument.
- **Panning:** toggles left/right output.
- **Pitch:** fine pitch.

# Commodore VIC instrument editor

---

the VIC instrument editor consists of these macros:

- **Volume**: volume sequence.
- **note**: global! affects entire chip.
- **Arpeggio**: pitch sequence.
- **On/Off**: enable/disable channel output.
- **Waveform**: square wave distortion type sequence.
- **Pitch**: "fine" pitch.

# Virtual Boy instrument editor

---

the Virtual Boy instrument editor contains three tabs: Virtual Boy, Wavetable and Macros.

## Virtual Boy

---

- **Set modulation table:** when enabled, playing this instrument on channel 5 will write to the modulation table.
- **Modulation table:** this allows you to define a waveform for frequency modulation.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Noise Length:** sets the noise length. higher values result in shorter noise.
- **Waveform:** wavetable sequence.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of waveform.

# VRC6 instrument editor

---

the VRC6 (regular) instrument editor consists of two tabs.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

note that using samples on VRC6 is CPU expensive!

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Duty**: specifies duty cycle for pulse wave channels.
- **Pitch**: fine pitch.

## VRC6 (saw) instrument editor

---

this channel has its own instrument type, a thing in Furnace that was decided as a compromise during a debate.

the only differences from this instrument type compared to the regular one are:

- the lack of a Sample tab.
- it has a volume range of 0-63 instead of 0-15.
- it lacks a duty cycle macro.
- if you come from FamiTracker, this may seem strange to you, but it isn't.

# Watara Supervision instrument editor

---

the Watara Supervision instrument editor consists of two tabs: one controlling sample channel as signments and macro tab containing several macros.

## Sample

---

for sample settings, see the the [Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of a "Use wavetable" option, and the presence of a "Use sample" one.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Duty/Noise:** pulse wave duty cycle or noise mode sequence.
- **Noise/PCM Pan:** output for left and right channels for noise and PCM channels.
- **Pitch:** fine pitch.

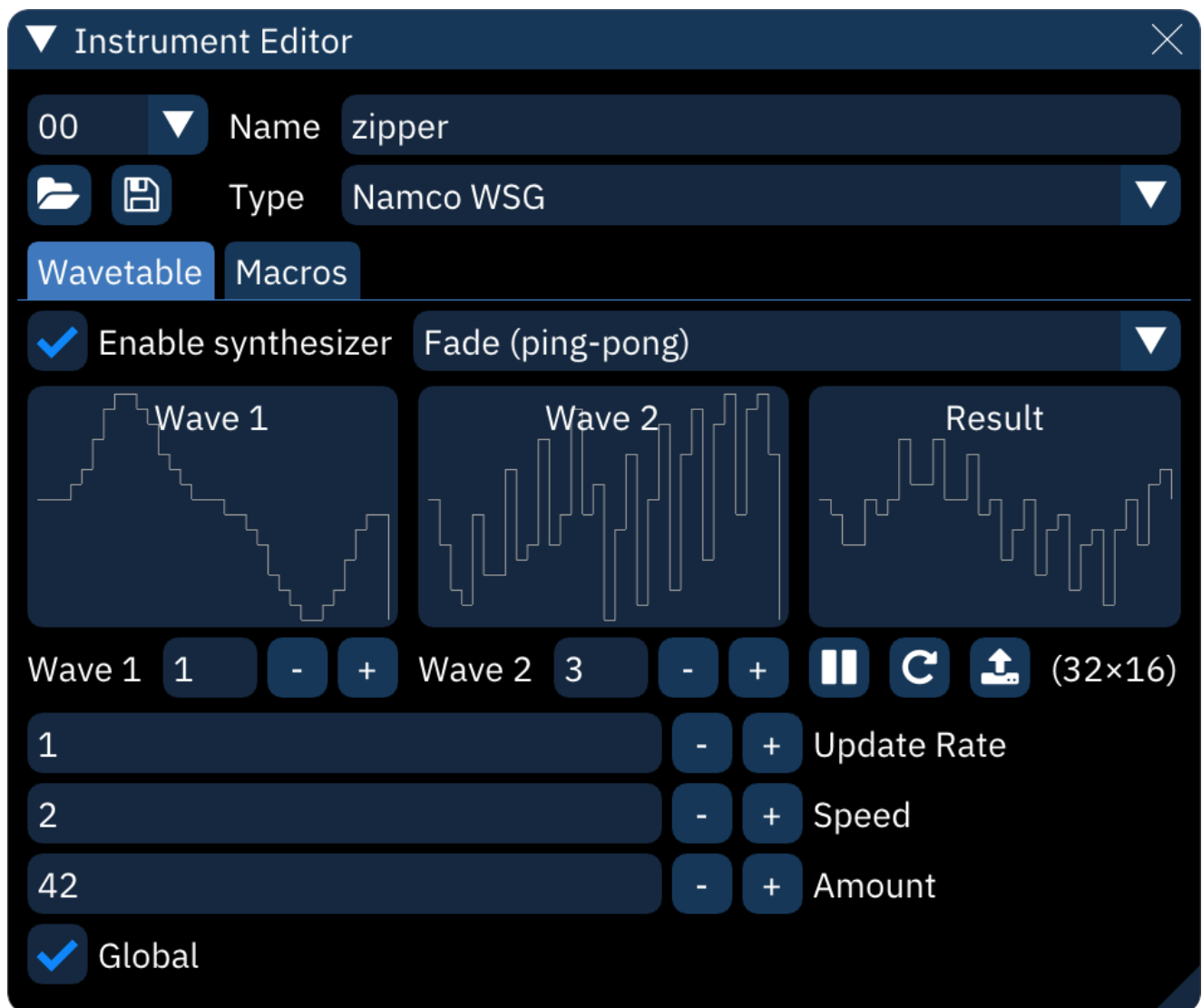


# wavetable synthesizer

within the "Wavetable" tab of the instrument editor, Furnace allows you to modulate or combine 1 or 2 waves to create unique "animated" sounds. think of it like a VST or a plugin, as it's basically an extension of regular wavetable soundchips that still allow it to run on real hardware.

this is accomplished by selecting a wave or two, a mode, and adjusting the settings as needed until you come up with a sound that you like, without taking up a load of space. this allows you to create unique sound effects or instruments, that, when used well, almost sound like they're Amiga samples.

unfortunately, on some chips like the HuC6280, you cannot use the wavetable synth to animate waveforms and have them sound smooth, as the chip resets the channel's phase when a wave form is changed while the channel is playing. on certain frequencies, this can be avoided, but not on most, unfortunately.



input waveforms should match the size of the wavetable or unexpected results may occur.

- **Enable synthesizer:** must be on for the rest of this to work.

- synthesizer type: selects the synthesis algorithm.
- waveform displays.
- **Wave 1**: selects input waveform.
- this will turn yellow to indicate that a Waveform macro is set.
- **Wave 2**: selects second input waveform. only appears when a dual-waveform synthesizer is selected.
- **Pause preview**: toggles live waveform preview.
- **Restart preview**: restarts preview from initial state.
- **Copy to new wavetable**: copies the currently displayed output waveform into the wavetable as a new entry.
- (width/height): size of wavetable.
- **Update Rate**: time in ticks between waveform changes.
- **Speed**: rate of change with each update.
- **Amount**: strength of synthesizer function.
- **Power**: only appears when synthesizer type is "Phase Modulation".
- **Global**:
- if disabled, each note resets the synthesizer to the start.
- if enabled, synthesis continues unbroken from note to note.

# WonderSwan instrument editor

---

the WonderSwan instrument editor contains three tabs: Sample, Wavetable and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

only on channel 2!

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Macros

---

these are similar to PC Engine, with some differences.

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Noise**: set noise size.
- only on channel 4.
- **Waveform**: wave sequence.
- **Pitch**: fine pitch.
- **Phase Reset**: trigger restart of waveform.

# Namco WSG instrument editor

---

the Namco WSG/C15/C30 instrument editor consists of two tabs: Wavetable and Macros.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Macros

---

- **Volume**: volume sequence.
- **Arpeggio**: pitch sequence.
- **Noise**: specifies noise pitch.
- only applicable for Namco C30.
- **Waveform**: specifies wavetable sequence.
- **Panning (left)**: output level of left channel.
- Namco C30 only.
- **Panning (right)**: output level of right channel.
- Namco C30 only.
- **Pitch**: fine pitch.

# X1-010 instrument editor

---

X1-010 instrument editor contains three tabs: Sample, Wavetable and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

the only differences are the lack of an "Use wavetable" option, and the presence of a "Use sample" one.

there also is a "Sample bank slot" setting, but I think that does nothing for now.

## Wavetable

---

this allows you to enable and configure the Furnace wavetable synthesizer. see [this page](#)<sup>205</sup> for more information.

## Macros

---

- **Volume:** volume levels sequence.
- **Arpeggio:** pitch sequence.
- **Waveform:** waveform selection sequence.
- **Panning (left):** output level for left channel.
- **Panning (right):** output level for right channel.
- **Pitch:** fine pitch.
- **Envelope Mode:** sets up envelope. the way it works is kind of complicated and even I don't understand how it works, so it's not documented for now.
- **Envelope:** specifies which wavetable should be used for envelope.
- **AutoEnv Num:** sets the envelope to the channel's frequency multiplied by numerator.
- **AutoEnv Den:** sets the envelope to the channel's frequency divided by denominator.
- the X1-010 hardware envelope is considerably slower than AY's. beware!
- these two must be set in order for AutoEnv to work!

# YMZ280B instrument editor

---

the YMZ280B instrument editor contains three tabs: Sample and Macros.

## Sample

---

for sample settings, see [the Sample instrument editor](#)<sup>183</sup>.

## Macros

---

- **Volume:** volume sequence.
- **Arpeggio:** pitch sequence.
- **Panning:** stereo panning sequence.
- **Pitch:** fine pitch.
- **Phase Reset:** trigger restart of sample.

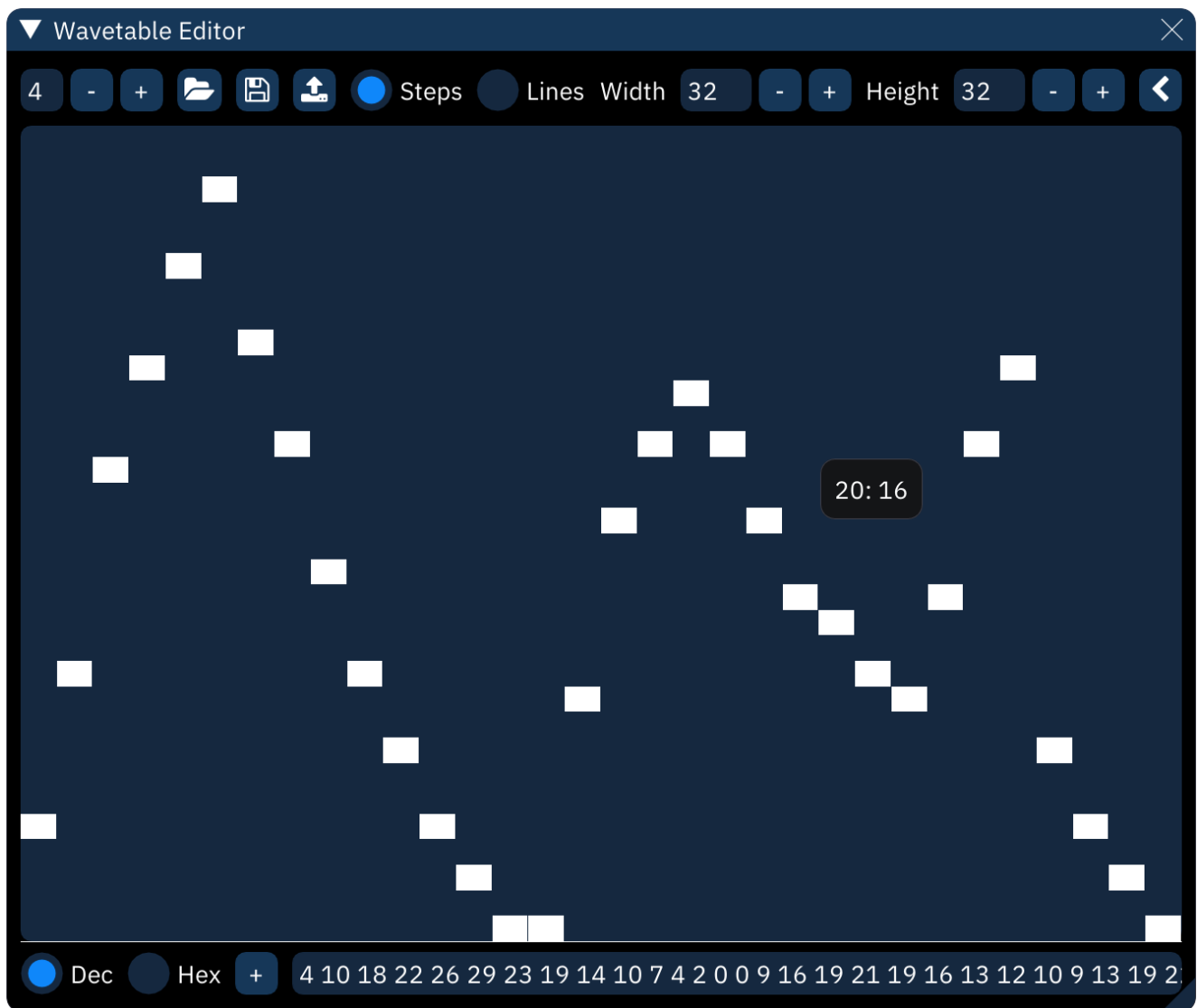
# wavetables

---

wavetable chips, in context of Furnace, are sound generators that operate on extremely short, looping sample streams. by extremely short, usually no more than 256 samples. this amount of space is nowhere near enough to store an actual sampled sound, but it allows certain amount of freedom to define a waveform shape.

each chip has its own maximum size, shown in the following table. if a larger wave is defined for these chips, it will be scaled to fit within the constraints of the chips. some of these don't work well with the wavetable synthesizer (described below); these systems are marked in the "notes" column.

SYSTEM	WIDTH	HEIGHT	NOTES
Bubble System	32	16	
Game Boy	32	16	phase reset on waveform change (clicking)
SM8521	32	16	
Namco WSG	32	16	RAM only
WonderSwan	32	16	
Namco 163	≤240	16	limits differ depending on channel count
SNES	≤256	16	
PC Engine	32	32	phase reset on waveform change (clicking)
Virtual Boy	32	64	
FDS	64	64	
Konami SCC	32	256	
Seta X1-010	128	256	
Amiga	≤256	256	
### wavetable editor			



controls across the top line:

- waveform number: the - and + buttons step through the list.
- open: opens a file selector to choose the file to open.
- save: opens a file selector to choose the file to save to.
- right-clicking brings up a menu:
  - **save as .dmw...**: saves the selected wavetable in DefleMask format.
  - **save raw...**: saves the selected wavetable as raw data.
- create sample from wavetable: copies wavetable to a new looped sample.
- **Steps**: view waveform as discrete blocks.
- **Lines**: view waveform as a continuous line.
- **Width**: length of the waveform data. maximum is 256.
- **Height**: height of the waveform. maximum is 256.
- < / >: show/hide waveform utilities (described below).

waveform display:

- the waveform is directly editable with the mouse.
- hovering will display a tooltip with the waveform position and value.



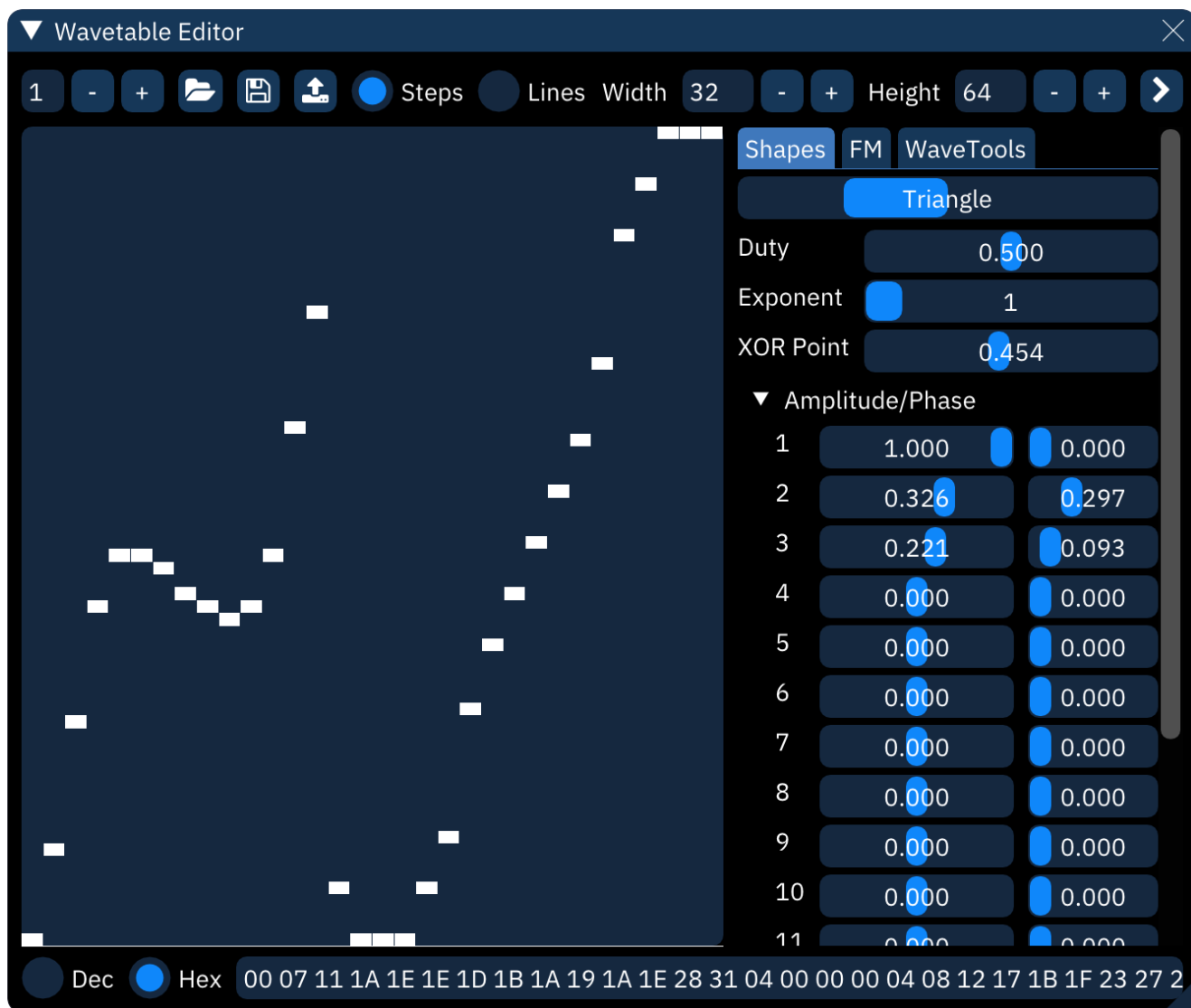
controls across the bottom line:

- **Dec**: view text input as decimal.
- **Hex**: view text input as hexadecimal.
- + / ±: toggle text input as unsigned/signed. also adjusts waveform display.
- text input: waveform data as an editable numeric sequence. also called "MML stream".

## waveform utilities

these provide different ways of creating or altering a waveform.

### Shapes

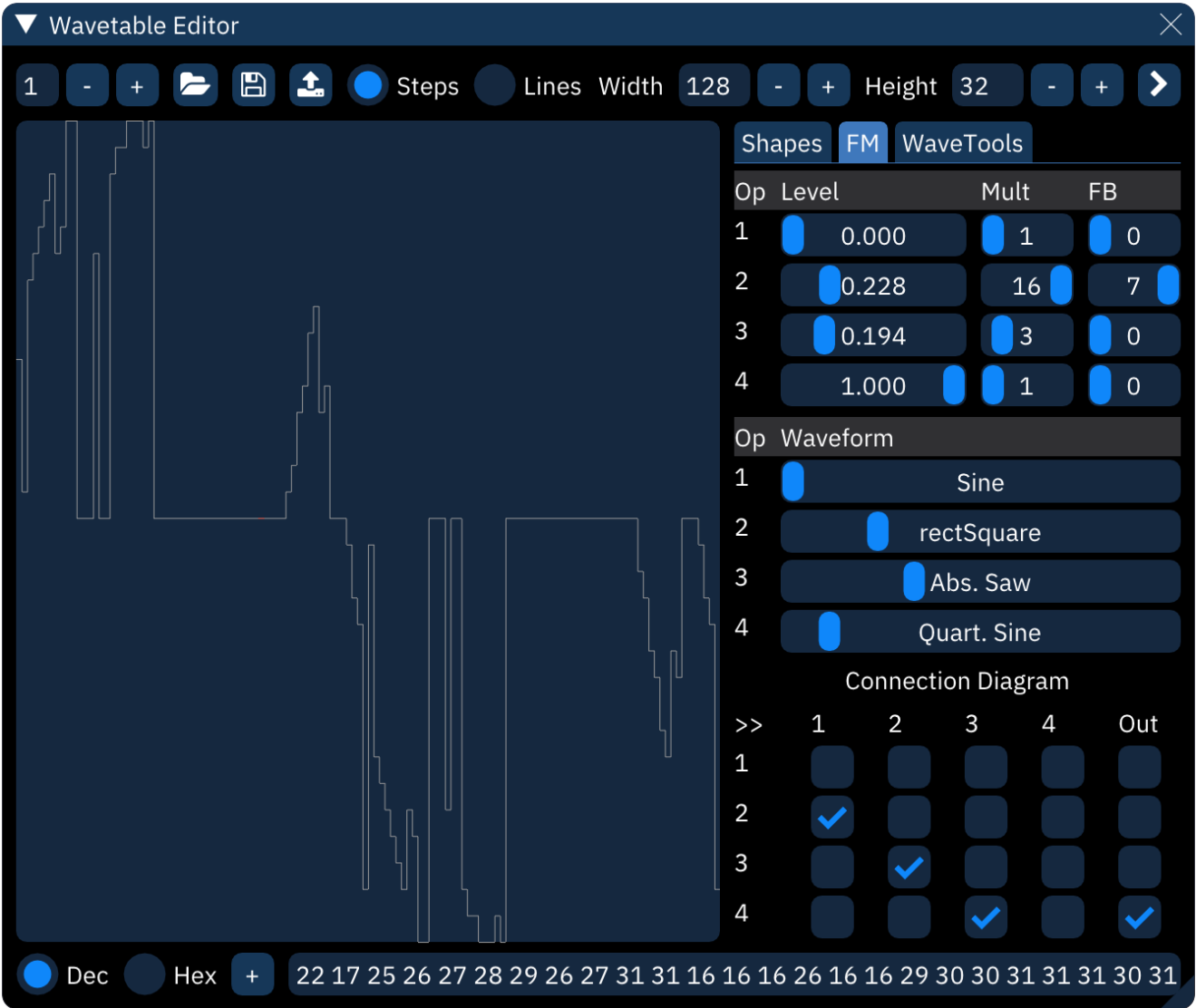


this creates a waveform by adding together a few predefined basic wave shapes.

- **shape**: select shape from sine, triangle, saw, and square.
- **Duty**: only affects pulse waves, determining their width.
- **Exponent**: applies an exponent (power) to the waveform ( $^2$ ,  $^3$  and so on).
- **XOR Point**: determines the point where the waveform gets inverted.
- **Amplitude/Phase**: add together up to 16 instances of the shape.

- **Amplitude:** height of the shape.
- **Phase:** position along the shape. for example, 0.250 starts the shape a quarter of the way along.

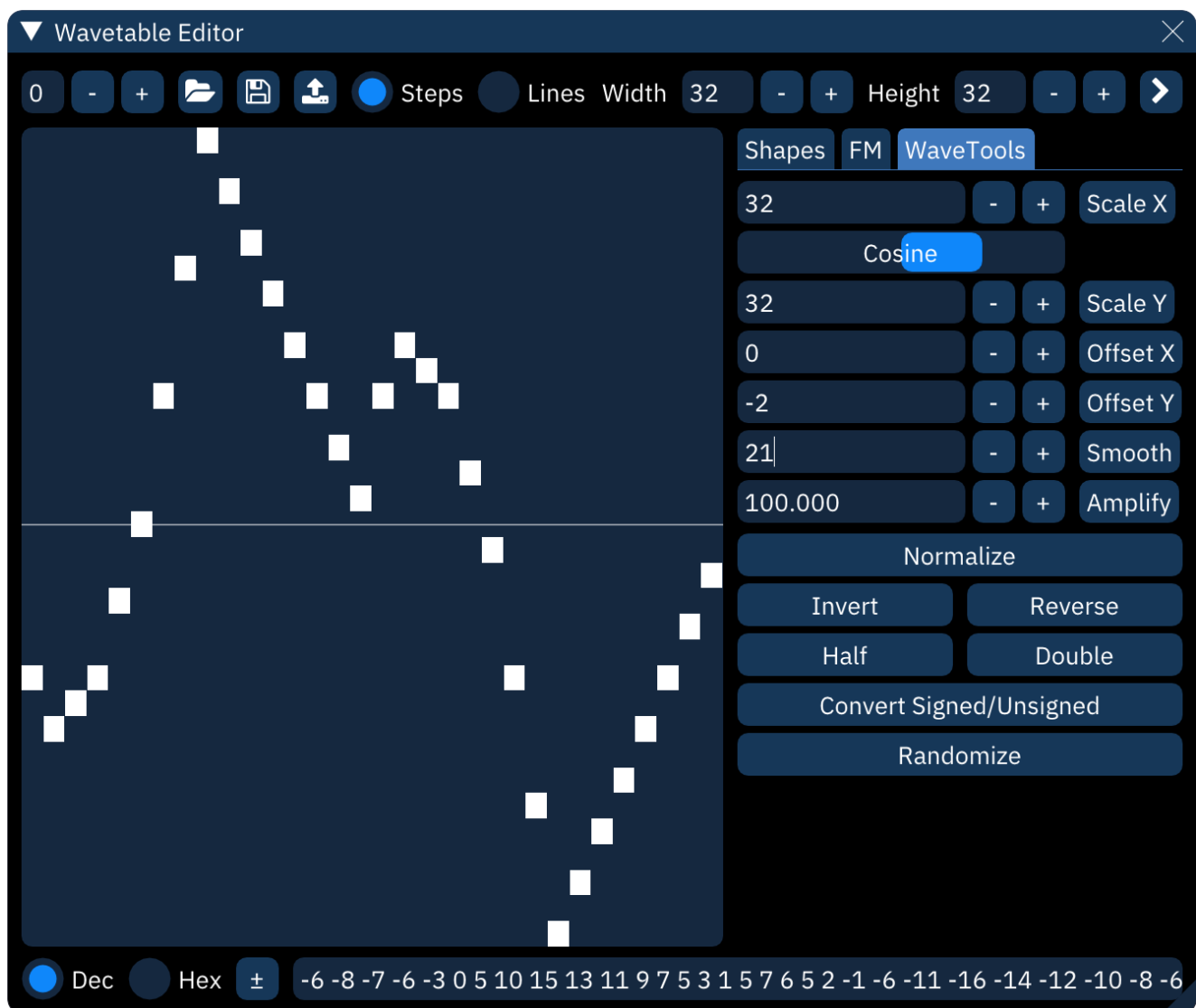
FM



this creates a waveform using frequency modulation synthesis with up to four operators.

you can set carrier/modulation levels, frequency multipliers, connections between operators and FM waveforms of these operators.

# WaveTools



these are useful editing tools to fine-tune the waveform:

- **Scale X**: stretches the waveform to a new length.
- **interpolation method**: filters the waveform when stretching. choose from none, linear, cosine, and cubic interpolation.
- **Scale Y**: resizes the waveform to a new height. it will clip at the top and bottom.
- **Offset X**: slides the the waveform forward or back. it will wrap around.
- **Offset Y**: slides the waveform up or down. it will clip at the top and bottom.
- **Smooth**. averages values in the waveform.
- **Amplify**. changes the volume of the waveform. it will clip at the top and bottom.
- **Normalize**: stretches waveform to maximum within the wavetable height.
- **Invert**: flips waveform vertically.
- **Reverse**: flips waveform horizontally.
- **Half**: halves the waveform's frequency by stretching its first half to fill the waveform length.
- **Double**: doubles the waveform's frequency by squashing it to half length then repeating it.
- **Convert Signed/Unsigned**. worth trying if an imported wave sounds corrupted.
- **Randomize**: generate a completely random waveform.

# samples

---

in the context of Furnace, a sound sample (usually just referred to as a sample) stores a sound.

in Furnace, these samples can be generated by importing a .wav file.

## supported chips

---

the following sound chips have sample support:

- NES/Ricoh 2A03 (with DPCM support and only on channel 5)
- Sega Genesis/YM2612 (channel 6 only)
- PC Engine/TurboGrafx-16
- Amiga
- SegaPCM
- YM2608 (ADPCM channel only)
- YM2610(B) (ADPCM channels only)
- Seta/Allumer X1-010
- Atari Lynx
- MSM6258
- MSM6295
- QSound
- ZX Spectrum 48K (1-bit overlay)
- RF5C68
- SNES
- WonderSwan (channel 2 only)
- Sound Unit
- VERA (last channel only)
- Y8950 (last channel only)
- Konami K007232
- Konami K053260
- Irem GA20
- Ensoniq OTTO/ES5506
- Yamaha PCMD8/YMZ280B
- MMC5 (last channel only)
- VRC6 (software!)
- AY-3-8910 (software!)
- AY8930 (software!)
- Namco C140
- Namco C219

## using samples

---

the simplest path to using a sample is:

- in the sample list, use the "Open" button (folder icon) to load the sample.
- double-click the sample in the list to open it in the sample editor.
- click the "Create instrument from sample" button (upload icon, to the left of "Zoom").

- use the created instrument in the track.

## compatible sample mode (LEGACY)

---

**use of this mode is discouraged in favor of Sample type instruments.**

effect 17xx enables/disables compatible sample mode where supported (e.g. on Sega Genesis or PC Engine).

in this mode, samples are mapped to notes in an octave from C to B, allowing you to use up to 12 samples.

if you need to use more samples, you may change the sample bank using effect EBxx.

## notes

---

due to limitations in some of those sound chips, some restrictions exist:

- Amiga: maximum frequency is 31469Hz, but anything over 28867 will sound glitchy on hardware. sample lengths and loop will be set to an even number, and your sample can't be longer than 131070.
- NES: if on DPCM mode, only a limited selection of frequencies is available.
- SegaPCM: your sample can't be longer than 65535, and the maximum frequency is 31.25KHz.
- QSound: your sample can't be longer than 65535, and the loop length shall not be greater than 32767.
- ADPCM-A: no looping supported. your samples will play at around 18.518KHz.
- ADPCM-B/YM2608: no loop position supported (only entire sample), and the maximum frequency is 55.555KHz.
- MSM6258/MSM6295: no arbitrary frequency.
- ZX Spectrum Beeper: your sample can't be longer than 2048, and it always plays at ~55KHz.
- Seta/Allumer X1-010: frequency resolution is terrible in the lower end. your sample can't be longer than 131072.
- C219: sample lengths and loop will be set to an even number, and your sample can't be longer than 131070.

furthermore, many of these chips have a limited amount of sample memory. check memory usage in window > statistics.

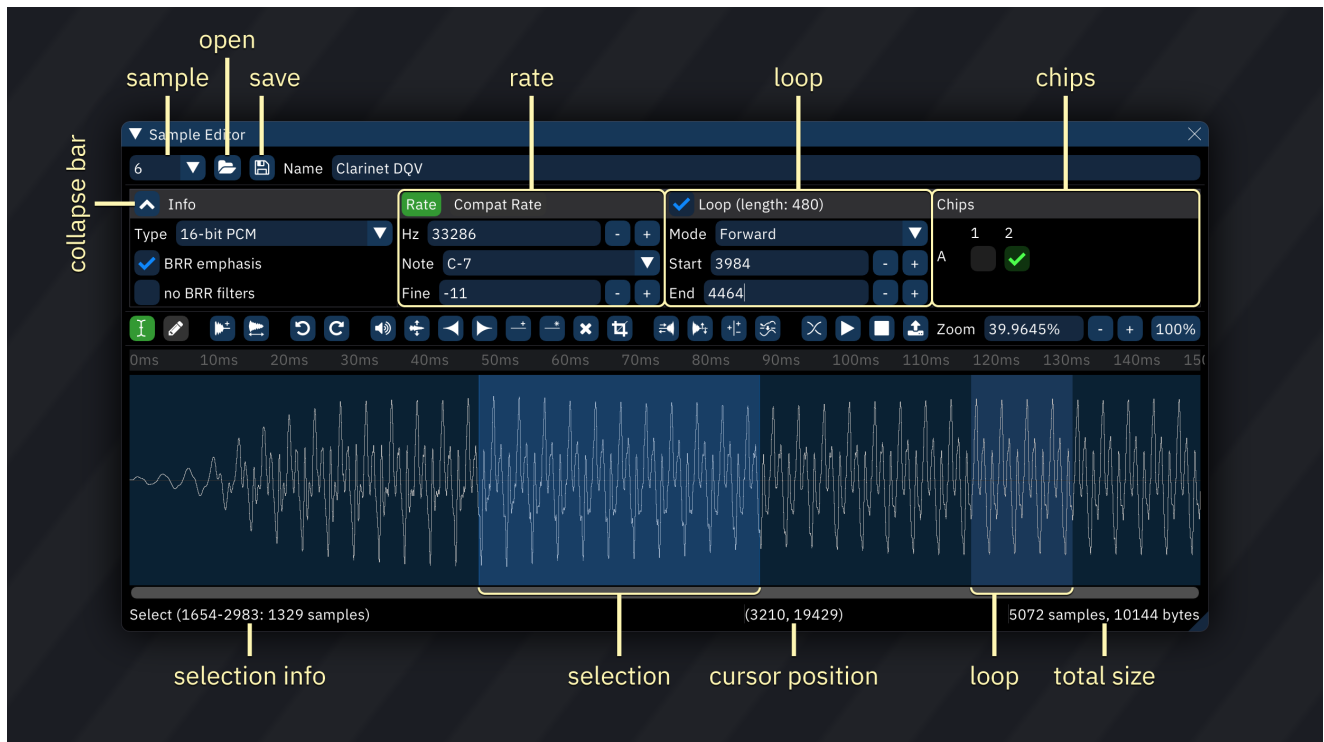
## the sample editor

---

you can edit your samples in Furnace's sample editor, which can be accessed by clicking on window (at the top of the screen) then clicking on sample editor, or by double-clicking a sample in the sample list.

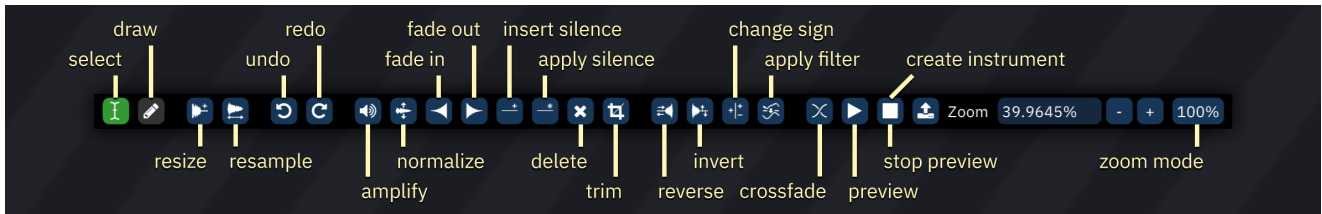
in there, you can modify certain data pertaining to your sample, such as the:

- volume of the sample in percentage, where 100% is the current level of the sample (note that you can distort it if you put it too high)
- the sample rate.
- what frequencies to filter, along with filter level/sweep and resonance options (much like the C64)
- and many more.



- top-left drop-down box: sample slot.
- **Open**: replaces current sample.
- right-clicking brings up a menu:
  - **import raw...**: brings up a file selector, then presents a dialog to choose the format of the selected file.
  - **import raw (replace)...**: same as above, but instead of adding it to the sample list, it replaces the currently selected sample.
- **Save**: saves current sample to disk.
- right-clicking brings up a menu:
  - **save raw...**: brings up a file selector, then saves the sample as raw data.
- **Name**: name in sample list.
- button to left of **Info**: collapses and expands the info section.
- **Type**: sample format.
- only 8-bit and 16-bit PCM samples are editable.
- selecting a format converts the sample data.
- **BRR emphasis**: boosts higher frequencies to compensate for the SNES low-pass filter.
- should not be enabled for BRR-type samples.
- only appears when applicable.
- **8-bit dither**: applies dithering to samples meant to play back at 8-bit resolution.
- only appears when applicable.
- **no BRR filters**: when encoding to BRR, only use a "4-bit mode" with block filter set to 0.
- this allows usage of sample offset effects on SNES.
- only appears when applicable.
- **Rate**: switches to normal rate values.
- **Compat Rate**: switches to DefleMask-compatible rate values for sample mapping.
- **use of this is discouraged!**
- **Hz**: base frequency of sample played at C-4.
- **Note**: note corresponding to Hz.
- **Fine**: fine tuning. ranges from -64 to 63, which maps to -1 to almost +1 semitone.
- **Loop**: enable or disable sample loop. only on supported chips.

- **Mode:** direction of loop. backward and ping pong loops are only natively available on some chips.
- **Start:** start of loop.
- **End:** end of loop.
- **Chips:** set assignment to chips and sample banks.
- sample will only be uploaded to selected chips.
- columns correspond to chips in use.
- rows correspond to sample banks.



- **Edit mode: Select:** cursor selects portion of sample.
- **Edit mode: Draw:** cursor draws over wave.
- **Resize:** stretches sample. pops up a dialog to type new length.
- **Resample:** stretches sample. pops up a dialog box:
  - **Rate:** new sample rate.
  - **0.5x:** halves sample rate.
  - **==:** returns to original sample rate.
  - **2.0x:** doubles sample rate.
  - **Factor:** multiplier of original sample rate.
- **Filter:** selects interpolation filter for resampling.
- **Undo:** undoes previous edit.
- **Redo:** redoes undone edit.
- **Amplify:** changes amplitude of selection. pops up a dialog to type amount.
- **Normalize:** adjusts amplitude of selection to maximum without clipping.
- **Fade in:** ramp amplitude of selection from 0 to original.
- **Fade out:** ramp amplitude of selection from original to 0.
- **Insert silence:** inserts silence. pops up a dialog to type length.
- **Apply silence:** reduces amplitude of selection to 0.
- **Delete:** removes selection.
- **Trim:** removes all but selection.
- **Reverse:** reverses direction of selection.
- **Invert:** flips selection "vertically".
- **Signed/unsigned exchange:** reinterprets selection data as being of the opposite sign.
  - if a sample sounds fine elsewhere but is distorted on import, it may have been interpreted as signed when it should be unsigned, or vice versa; this will correct that.
- **Apply filter:** filters the selection. pops up a dialog box:
  - **Sweep (2 frequencies):** if checked, the "Frequency" slider is replaced with "From" and "To" sliders, and the filter cutoff sweeps between the two.
    - **Frequency:** filter cutoff frequency.
    - **From:** filter cutoff frequency at start of selection.
    - **To:** filter cutoff frequency at end of selection.
  - **Resonance:** emphasizes frequencies around filter cutoff.
  - **Power:** number of times resonance is applied.
  - **Low-pass:** amount to attenuate everything above cutoff.
  - **Band-pass:** amount to attenuate everything outside cutoff.
  - **High-pass:** amount to attenuate everything below cutoff.

- **Crossfade loop points:** applies a "fade" between the loop's starting point and the end.
- **Number of samples:** how many samples in the loop region to take into account for crossfade.
- **Linear <-> Equal power:** the curve used to crossfade.
- **Preview sample:** plays sample at base frequency.
- **Stop sample preview:** stops preview.
- **Create instrument from sample:** creates a new instrument with its sample set to the current sample.
- **Zoom:** shows and sets sample view zoom level.
- **Zoom mode:** switches between "Auto" (entire sample fits in window) and "100%" (each horizontal pixel represents one sample point).

in the sample viewer:

- left-click and drag to select a region of the sample.
- right-click to display a menu:
- **cut:** puts the selection in the sample clipboard and deletes it from the sample.
- **copy:** copies the selection into the sample clipboard.
- **paste:** inserts the sample clipboard at the start of the selection.
- **paste (replace):** replaces the selection with the sample clipboard.
- **paste (mix):** mixes the sample clipboard into the existing sample, beginning at the start of the selection.
- **set loop to selection:** changes loop region to match selection.
- **create wavetable from selection:** copies the selection into a new wavetable entry.



# systems

---

this is a list that contains some of the systems that Furnace supports, as shown in the New File dialog. each chip links to a page with information and a list of supported effects.

some systems have alternate chips, such as the Sega Genesis having a YM2612 or YM3438 depending on the model. this list shows the default configuration.

- **Sega Genesis:** [YM3438](#)<sup>346</sup>, [SN76489](#)<sup>298</sup>
- **Sega Genesis (with Sega CD):** [YM3438](#)<sup>346</sup>, [SN76489](#)<sup>298</sup>, [RF5C164](#)<sup>287</sup>
- **Sega Master System:** [SN76489](#)<sup>298</sup>
- **Sega Master System (with FM expansion):** [SN76489](#)<sup>298</sup>, [YM2413](#)<sup>271</sup>
- **NES:** [2A03](#)<sup>264</sup>
- **Famicom with Konami VRC6:** [2A03](#)<sup>264</sup>, [VRC6](#)<sup>322</sup>
- **Famicom with Konami VRC7:** [2A03](#)<sup>264</sup>, [VRC7](#)<sup>271</sup>
- **Famicom with MMC5:** [2A03](#)<sup>264</sup>, [MMC5](#)<sup>255</sup>
- **Famicom with Sunsoft 5B:** [2A03](#)<sup>264</sup>, [5B](#)<sup>228</sup>
- **Famicom with Namco 163:** [2A03](#)<sup>264</sup>, [N163](#)<sup>260</sup>
- **Famicom Disk System:** [2A03](#)<sup>264</sup>, [FDS](#)<sup>247</sup>
- **Game Boy:** [Game Boy](#)<sup>249</sup>
- **SNES:** [SNES](#)<sup>300</sup>
- **NEC PC Engine/TurboGrafx-16:** [HuC6280](#)<sup>276</sup>
- **Commodore VIC-20:** [VIC](#)<sup>319</sup>
- **Commodore 64 (6581 SID):** [MOS 6581](#)<sup>236</sup>
- **Commodore 64 (8580 SID):** [MOS 8580](#)<sup>236</sup>
- **Amiga:** [Amiga](#)<sup>226</sup>
- **Arcade (YM2151 and SegaPCM):** [YM2151](#)<sup>328</sup>, [SegaPCM](#)<sup>290</sup>
- **Capcom CPS-1:** [YM2151](#)<sup>328</sup>, [MSM6295](#)<sup>258</sup>
- **Capcom CPS-2 (QSound):** [QSound](#)<sup>286</sup>
- **Neo Geo CD:** [YM2610](#)<sup>338</sup>
- **Neo Geo CD (extended channel 2):** [YM2610](#)<sup>338</sup>
- **Neo Geo Pocket:** [T6W28](#)<sup>308</sup>, [DAC](#)<sup>239</sup>
- **Atari 2600/7800:** [TIA](#)<sup>310</sup>

- **Atari 800:** [POKEY](#)<sub>282</sub>
- **Konami Bubble System:** [AY-3-8910](#)<sub>228</sub> / 2, [Konami WSG](#)<sub>233</sub>
- **Sharp X68000:** [YM2151](#)<sub>328</sub>, [MSM6258](#)<sub>257</sub>
- **PC + Sound Blaster Pro:** [YM3812](#)<sub>268</sub> / 2, [DAC](#)<sub>239</sub>, [PC Speaker](#)<sub>278</sub>
- **MSX:** [AY-3-8910](#)<sub>228</sub>
- **MSX + SCC:** [YM2149\(F\)](#)<sub>228</sub>, [SCC](#)<sub>289</sub>
- **Commander X16 (VERA only):** [VERA](#)<sub>318</sub>
- and many, many more!

## chips

---

this is the full list of chips that Furnace supports.

- [2A03](#)<sub>264</sub>
- [5E01](#)<sub>224</sub>
- [Amiga](#)<sub>226</sub>
- [AY-3-8910/8914/YM2149\(F\)/Sunsoft 5B](#)<sub>228</sub>
- [Microchip AY8930](#)<sub>230</sub>
- [Bifurcator](#)<sub>232</sub>
- [MOS 6581/8580 \(SID\)](#)<sub>236</sub>
- [Dave](#)<sub>240</sub>
- [Ensoniq ES5506](#)<sub>242</sub>
- [Konami SCC](#)<sub>289</sub>
- [FDS](#)<sub>247</sub>
- [Game Boy](#)<sub>249</sub>
- [Game Boy Advance](#)<sub>251</sub>
- [Generic PCM DAC](#)<sub>239</sub>
- [Irem GA20](#)<sub>248</sub>
- [Bubble System WSG](#)<sub>233</sub>
- [K007232](#)<sub>252</sub>
- [K053260](#)<sub>253</sub>
- [Lynx](#)<sub>254</sub>
- [MMC5](#)<sub>255</sub>
- [MSM5232](#)<sub>256</sub>
- [MSM6258](#)<sub>257</sub>
- [MSM6295](#)<sub>258</sub>
- [Namco 163](#)<sub>260</sub>
- [Namco C140](#)<sub>234</sub>
- [Namco WSG/C15/C30](#)<sub>262</sub>
- [Nintendo DS](#)<sub>263</sub>
- [HuC6280](#)<sub>276</sub>
- [PC Speaker](#)<sub>278</sub>
- [PET](#)<sub>280</sub>
- [Pokémon Mini](#)<sub>281</sub>
- [POKEY](#)<sub>282</sub>
- [PowerNoise](#)<sub>284</sub>

- PV-1000<sup>285</sup>
- QSound<sup>286</sup>
- RF5C68/RF5C164<sup>287</sup>
- SAA1099<sup>288</sup>
- SegaPCM<sup>290</sup>
- SID2<sup>291</sup>
- SID3<sup>293</sup>
- SM8521<sup>297</sup>
- SN76489/Sega PSG<sup>298</sup>
- SNES<sup>300</sup>
- tildearrow Sound Unit<sup>306</sup>
- T6W28<sup>308</sup>
- TED<sup>309</sup>
- TIA<sup>310</sup>
- VERA<sup>318</sup>
- VIC<sup>319</sup>
- Virtual Boy<sup>320</sup>
- VRC6<sup>322</sup>
- Watara Supervision<sup>323</sup>
- WonderSwan<sup>324</sup>
- X1-010<sup>326</sup>
- Y8950, YM3526, YM3812, YMF262 and YMF278 (OPL)<sup>268</sup>
- ESFM<sup>244</sup>
- VRC7 and YM2413 (OPLL)<sup>271</sup>
- YM2414 (OPZ)<sup>273</sup>
- YM2151 (OPM)<sup>328</sup>
- YM2203 (OPN)<sup>330</sup>
- YM2608 (OPNA)<sup>334</sup>
- YM2610 (OPNB)<sup>338</sup>
- YM2610B (OPNB2)<sup>342</sup>
- YM2612/YM3438 (OPN2)<sup>346</sup>
- YMZ280B<sup>351</sup>
- ZX Spectrum Beeper<sup>352</sup>

Furnace also reads .dmf files with the [Yamaha YMU759](#)<sup>350</sup> system, but...

# 5E01

---

a fantasy sound chip created by Euly, based on the Ricoh 2A03, with some improvements:

- a 37.5% duty cycle,
- 32 noise pitches instead of 16, and
- triangle channel becomes a wave channel, with four available waveforms: triangle, saw, sine and square.

## effects

---

- 11xx: **write to delta modulation counter.** range is 00 to 7F.
- this may be used to attenuate the triangle and noise channels; at 7F, they will be at about 57% volume.
- will not work if a sample is playing.
- 12xx: **set duty cycle/noise mode/waveform of channel.**
- may be 0 to 3 for the pulse channels:
  - 0: 12.5%
  - 1: 25%
  - 2: 37.5%
  - 3: 50%
- may be 0 or 1 for the noise channel:
  - 0: long (15-bit LFSR, 32767-step)
  - 1: short (9-bit LFSR, 93-step)
- may be 0 to 3 for the wave channel:
  - 0: triangle
  - 1: saw
  - 2: square
  - 3: sine
- 13xy: **setup sweep up.**
- x is the time.
- y is the shift.
- set to 0 to disable it.
- 14xy: **setup sweep down.**
- x is the time.
- y is the shift.
- set to 0 to disable it.
- 15xx: **set envelope mode.**
- 0: envelope + length counter. volume represents envelope duration.
- 1: length counter. volume represents output volume.
- 2: looping envelope. volume represents envelope duration.
- 3: constant volume. default value. volume represents output volume.
- pulse and noise channels only.
- you may need to apply a phase reset (using the macro) to make the envelope effective.
- 16xx: **set length counter.**
- see [NES<sub>264</sub>](#) for more information.
- this will trigger phase reset.
- 17xx: **set frame counter mode.**

- 0: 4-step.
- 1: 5-step.
- 18xx: **set PCM channel mode.**
- 00: PCM (software).
- 01: DPCM (hardware).
- when in DPCM mode, samples will sound muffled (due to its nature), available pitches are limited, and loop point is ignored.
- 19xx: **set triangle linear counter.**
- 00 to 7F set the counter.
- 80 and higher halt it.
- 20xx: **set DPCM frequency.**
- only works in DPCM mode.

## info

---

this chip uses the [NES<sub>170</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **DPCM channel mode:** allows you to set which mode to use for the DPCM channel.
- DPCM: the default mode, playing 1-bit DPCM samples as supported by the hardware.
- PCM: this mode provides crisper samples by writing the delta counter directly. uses a lot of CPU time in console.

# Commodore Amiga

---

a computer with a desktop OS, lifelike graphics and 4 channels of PCM sound in 1985? no way!

in this very computer music trackers were born...

imported MOD files use this chip, and will set A-4 tuning to 436.

## amplitude/period modulation

---

Amiga has support for (rather primitive) amplitude and period (frequency) modulation. however, nobody has used this feature as it is rather useless, not well-documented and works in a complicated way.

Amiga sample playback is done by two chips: Paula (the one that you probably know) and Agnus (the one that actually feeds Paula with samples).

Agnus has several DMA (direct memory access) units which read from chip memory independent of the CPU. four of these DMA units are used for samples.

when DMA is enabled, Paula requests sample data from Agnus, and then plays these samples back.

there's a catch though. since the data bus is 16-bit, Paula requests **two** 8-bit samples at once! this explains why:

- the sample length registers are in words rather than bytes (thereby allowing samples up to 131070 in length)
- the maximum playback rate (31250Hz PAL; ~31469Hz NTSC) is two times the HBlank rate (Agnus fetches samples on HBlank, around 15625Hz on PAL or ~15734Hz on NTSC)

during normal sample playback, the first sample is output and then the second. afterwards, two more samples are fetched, and so on.

now, when amplitude or period modulation are enabled, things work differently.

the channel is silenced, and the two 8-bit samples are **treated as a big-endian 16-bit number**, which is then written to the next channel's volume or period.

in the case of amplitude modulation, only the second sample is significant because the volume register uses 7 bits (to represent 0 to 64 (65 to 127 are treated as 64)) and the other bits are ignored.

in the case of period modulation, both samples are significant. the first sample is the upper byte, and the second is the lower byte.

## effects

---

- 10xx: **toggle low-pass filter**. 0 turns it off and 1 turns it on.
- 11xx: **toggle amplitude modulation with the next channel**.
- does not work on the last channel.
- 12xx: **toggle period (frequency) modulation with the next channel**.
- does not work on the last channel.

- 13xx: **change wave.**
- only works when "Mode" is set to "Wavetable" in the instrument.

## info

---

this chip uses the [Generic Sample<sub>183</sub>](#) instrument editor.

- the maximum rate for sample playback is technically 31469Hz but anything higher than 28867Hz will sound glitchy on hardware.
- sample lengths and loop will be set to an even number.
- samples can't be longer than 131070.

## chip config

---

the following options are available in the Chip Manager window:

- **Stereo separation:** sets the amount of left/right separation.
- **Model:** allows you to change the chipset.
- Amiga 500 (OCS): has a low-pass filter on top of the user-selectable filter.
- Amiga 1200 (AGA): doesn't have the aforementioned low-pass filter.
- **Chip memory:** more chip memory means more space for samples.
- **PAL:** run the chip at PAL clock (3.54MHz) instead of NTSC (3.58MHz).
- **Bypass frequency limits:** when enabled, the ~31KHz frequency limit is disabled, allowing you to play higher notes.

# General Instrument

## AY-3-8910

---

this chip was used in many home computers (ZX Spectrum, MSX, Amstrad CPC, Atari ST, etc.), video game consoles (Intellivision and Vectrex), arcade boards and even slot machines!

it is a 3-channel square/noise/envelope sound generator. the chip's powerful sound comes from the envelope...

the AY-3-8914 variant was used in Intellivision, which is pretty much an AY with 4 level envelope volume per channel and different register format.

Furnace is capable of doing software sample playback on AY-3-8910, where all 3 channels can play 4-bit PCM samples (at the cost of a very high CPU usage) and utilize CPU timer effects, providing all kinds of modulation (PWM and envelope distortion) effects, again at the cost of a higher CPU usage. Songs utilizing timer effects may not sound as intended after VGM export.

## effects

---

- 20xx: **set channel mode.**
- 0: square
- 1: noise
- 2: square and noise
- 3: envelope
- 4: envelope and square
- 5: envelope and noise
- 6: envelope and square and noise
- 7: nothing
- 21xx: **set noise frequency.** range is 0 to 1F.
- 22xy: **set envelope mode.**
- x sets the envelope shape:
  - 0: \\_\_\_ decay
  - 4: /\_\_\_ attack once
  - 8: \\\ saw
  - 9: \\_\_\_ decay
  - A: /\ / inverse obelisco
  - B: \ ^ ^ ^ decay once
  - C: /// inverse saw
  - D: / ^ ^ ^ attack
  - E: /\ / obelisco
  - F: /\_\_\_ attack once
- if y is 1 then the envelope will affect this channel.
- 23xx: **set envelope period low byte.**
- 24xx: **set envelope period high byte.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**



- 29xy: **enable auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.
- 2Cxx: **set timer period offset.**
- bit 7 is the sign.
- 2Exx: **write to I/O port A.**
- this changes the port's mode to "write". make sure you have connected something to it.
- 2Fxx: **write to I/O port B.**
- this changes the port's mode to "write". make sure you have connected something to it.

## what is obelisco

---

it's a name I use for a spiky waveform that starts low.

its origin is a wavetable that comes in DefleMask that happens to be called Obelisco.

## AY derivative modes

---

AY-3-810 was an absurdly popular chip that was blessed with many third-party clones, licensed or not.

- the AY-3-8914 variant was used in Intellivision, which is pretty much an 8910 with 4 level envelope volume per channel and different register format.
- Yamaha YM2149 was an AY-3-8910 clone released in 1983. it's almost identical to AY with minor differences being: higher hardware envelope step resolution (16 vs 32), half-clock mode when voltage level is low, much stronger DC offset and cleaner, but softer output.
- Sunsoft 5B is YM2149 clone with half-clock mode forced on.

## info

---

this chip uses the [AY-3-8910<sup>124</sup>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Chip type:** changes the chip model.
- AY-3-8910: the original model.
- YM2149(F): Yamaha clone with higher envelope resolution and a different volume curve.
- Sunsoft 5B: a special model found in one of Sunsoft's Famicom cartridges.
- AY-3-8914: scrambled registers. has envelope volume control.
- **Stereo:** enable stereo output. channel 1 is output left, channel 2 is output center and channel 3 is output right.
- **Separation:** allows you to set left/right separation.
- **Half Clock divider:** pulls the half clock pin, running the chip at half the speed.
- only available in YM2149.

# Microchip AY8930

---

a backwards-compatible successor to the AY-3-8910, with increased volume resolution, duty cycle control, three envelopes and highly configurable noise generator.

sadly, this soundchip has only ever observed minimal success, and has remained rather obscure since.

it is known for being used in the Covox Sound Master, which didn't sell well either.

emulation of this chip in Furnace is now complete thanks to community efforts and hardware testing, which an MSX board called Darky has permitted.

Furnace is able to do software PCM on AY8930, where all 3 channels can play 5-bit PCM samples (at the cost of a very high CPU usage).

## effects

---

- 12xx: **set channel duty cycle.**
- 0: 3.125%
- 1: 6.25%
- 2: 12.5%
- 3: 25%
- 4: 50%
- 5: 75%
- 6: 87.5%
- 7: 93.75%
- 8: 96.875%
- 20xx: **set channel mode.** xx may be one of the following:
- 0: square
- 1: noise
- 2: square and noise
- 3: envelope
- 4: envelope and square
- 5: envelope and noise
- 6: envelope and square and noise
- 7: nothing
- 21xx: **set noise frequency.** xx is a value between 00 and FF.
- 22xy: **set envelope mode.**
- x sets the envelope shape, which may be one of the following:
  - 0: \\_\_\_ decay
  - 4: /\_\_\_ attack once
  - 8: \\\ saw
  - 9: \\_\_\_ decay
  - A: \/\/ inverse obelisco
  - B: \--- decay once
  - C: /// inverse saw
  - D: /--- attack
  - E: /\\/ obelisco

- F: / \_\_\_ attack once
- if y is 1 then the envelope will affect this channel.
- 23xx: **set envelope period low byte.**
- 24xx: **set envelope period high byte.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**
- 27xx: **set noise AND mask.**
- 28xx: **set noise OR mask.**
- 29xy: **enable auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.
- 2Cxx: **automatic noise frequency.**
- x sets the mode:
  - 0: disabled
  - 1: alter frequency
  - 2: alter frequency and OR mask
- y sets the offset.
- this can be used to make a pulse-width modulation (PWM) effect.
- 2Exx: **write to I/O port A.**
- this changes the port's mode to "write". make sure you have connected something to it.
- 2Fxx: **write to I/O port B.**
- this changes the port's mode to "write". make sure you have connected something to it.

## info

---

this chip uses the [AY8930<sub>125</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Stereo:** enable stereo output. channel 1 is output left, channel 2 is output center and channel 3 is output right.
- **Separation:** allows you to set left/right separation.
- **Half Clock divider:** pulls the half clock pin, running the chip at half the speed.

# Bifurcator

---

this is a fantasy sound chip which uses a unique sound generation method: logistic map iterations.

## effects

---

- 10xx: **set low byte of channel sample state.**
- 11xx: **set high byte of channel sample state.**
- 12xx: **set parameter low byte.**
- 13xx: **set parameter high byte.**

## info

---

this chip uses the [Bifurcator<sub>127</sub>](#) instrument editor.

# Bubble System WSG

---

a Konami-made 2 channel wavetable sound generator logic used on the Bubble System arcade board, configured with K005289, a 4-bit PROM and DAC.

however, the K005289 is just part of the logic used for pitch and wavetable ROM address. waveform select and volume control are tied with single AY-3-8910 IO for both channels. another AY-3-8910 IO is used for reading sound hardware status.

Furnace emulates this configuration as a "chip" with 32/16 wavetables.

## effects

---

- 10xx: **change wave.**

## info

---

this chip uses the [Konami SCC/Bubble System WSG<sub>184</sub>](#) instrument editor.

# Namco C140

---

the Namco C140 is a 24-channel custom PCM sound chip manufactured jointly by Fujitsu and Namco. it was first used in Namco System 2 arcade family starting in 1987.

this chip features:

- stereo soft panning
- accepts either raw 12-bit PCM or 8-bit -law compressed PCM samples
- 21.4 kHz sampling rate

## effects

---

none!

## info

---

this chip uses the [C140<sub>128</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Banking style:** change the sample bank scheme. used for VGM export.
- Namco System 2: 2MB available for samples.
- Namco System 21: 4MB available for samples.
- Raw: 16MB, but not supported by the VGM format!

# Namco C219

---

Namco C219 is a 16-channel PCM sound chip that's a lot like C140, but has a noise generator, surround effect and a different  $\gamma$ -law curve.

this chip features:

- stereo soft panning
- accepts either 8-bit PCM or proprietary 8-bit  $\gamma$ -law compressed PCM samples

## sample memory notice

---

this chip is rather unique when it comes to sample memory. be sure to read this notice.

the channels are in groups of four. a sample bank (128KB) may be selected for each group. if a sample that is on a different bank plays in a group, the group is switched to that bank, and other channels will be silenced.

## effects

---

- 11xx: **set noise mode.**
- 12xy: **set invert mode.**
- if x is 1 or higher, surround is enabled.
- if y is 1 or higher, invert is enabled.

## info

---

this chip uses the [C219<sub>129</sub>](#) instrument editor.

# Commodore 64

---

a home computer with a synthesizer-grade sound chip of which people took decades to master. three oscillators with four selectable waveforms, ring modulation, oscillator sync, multi-mode filter and ADSR envelope.

very popular in Europe and mostly due to the demoscene, which stretched the machine's limbs to no end.

two versions of aforementioned chip exist: 6581 (original chip) and 8580 (improved version with working waveform mixing and somewhat more consistent filter curves).

the 6581 has a hardware quirk which produces a DC output with its intensity being regulated by the global volume register. this register can be abused to produce a crude, virtual fourth 4-bit PCM channel.

Furnace supports this with the "Commodore 64 (SID 6581) with software PCM" system. the later 8580 chip fixed this problem, making such PCM nearly inaudible; while other PCM playback methods have been invented, Furnace does not support them at the moment.

## effects

---

- 10xx: **change wave.** the following values are accepted:
  - 00: nothing
  - 01: triangle
  - 02: saw
  - 03: triangle and saw
  - 04: pulse
  - 05: pulse and triangle
  - 06: pulse and saw
  - 07: pulse and triangle and saw
  - 08: noise
- 11xx: **set coarse cutoff.** xx may be a value between 00 and 64.
  - *this effect only exists for compatibility reasons, and its use is discouraged.*
  - use effect 4xxx instead.
- 12xx: **set coarse duty cycle.** xx may be a value between 00 and 64.
  - *this effect only exists for compatibility reasons, and its use is discouraged.*
  - use effect 3xxx instead.
- 13xx: **set resonance.** xx may be a value between 00 and 0F.
- 14xx: **set filter mode.** the following values are accepted:
  - 00: filter off
  - 01: low pass
  - 02: band pass
  - 03: low+band pass
  - 04: high pass
  - 05: band reject/stop/notch
  - 06: high+band pass
  - 07: all pass
- 15xx: **set envelope reset time.**



- this is the amount of ticks the channel turns off before a note occurs in order to reset the envelope safely.
- if xx is 0 or higher than the song speed, the envelope will not reset.
- 1Axx: **disable envelope reset for this channel.**
- 1Bxy: **reset cutoff:**
  - if x is not 0: on new note
  - if y is not 0: now
  - this effect is not necessary if the instrument's cutoff macro is absolute.
- 1Cxy: **reset duty cycle:**
  - if x is not 0: on new note
  - if y is not 0: now
  - this effect is not necessary if the instrument's duty macro is absolute.
- 1Exy: **change additional parameters.**
- *this effect only exists for compatibility reasons, and its use is discouraged.*
- x may be one of the following:
  - 0: attack (y from 0 to F)
  - 1: decay (y from 0 to F)
  - 2: sustain (y from 0 to F)
  - 3: release (y from 0 to F)
  - 4: ring modulation (y is 0 or 1)
  - 5: oscillator sync (y is 0 or 1)
  - 6: disable channel 3 (y is 0 or 1)
- 20xy: **set attack/decay.**
  - x is the attack.
  - y is the decay.
- 21xy: **set sustain/release.**
  - x is the sustain.
  - y is the release.
- 22xx: **pulse width slide up.**
  - xx is speed. if it is 00, the slide is stopped.
- 23xx: **pulse width slide down.**
  - xx is speed. if it is 00, the slide is stopped.
- 24xx: **filter cutoff slide up.**
  - xx is speed. if it is 00, the slide is stopped.
- 25xx: **filter cutoff slide down.**
  - xx is speed. if it is 00, the slide is stopped.
- 3xxx: **set duty cycle.** xxx range is 000 to FFF.
- 4xxx: **set cutoff.** xxx range is 000 to 7FF.

## info

---

this chip uses the [C64<sub>130</sub>](#) instrument editor.

## channel status

---

the following icons are displayed when channel status is enabled in the pattern view:

- channel is silent:



- it's not



- gate bit disabled



- gate bit disabled and test bit enabled



- test bit enabled



- ch3off enabled in filter mode

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Global parameter priority:** change the priority of macros which control global parameters, such as volume and filter.
- Left to right: process channels from 1 to 3. the last one to run macros will take effect.
- Last used channel: process channels from oldest to newest (since last note). the one which had the latest note on will take a effect.
- **Hard reset envelope:** configure the envelope parameters used during the short reset before a note.
- **Envelope reset time:** set how long will the pre-note reset last, in ticks.
- 0 disables reset, which prevents notes from triggering.
- 1 is short, but may exhibit SID envelope bugs.
- 2 is a good value.

the other options are for compatibility with old Furnace and DefleMask, so they won't be documented here.

# Generic PCM DAC

---

a sample channel, with freely selectable rate, mono/stereo and bit depth settings.

with it, you can emulate PCM DACs found in Williams arcade boards, Sound Blasters, MSX TurboR, Atari STe, NEC PC-9801-86, among others.

## effects

---

none yet.

## info

---

this chip uses the [Generic Sample<sub>183</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Output rate:** sets the output sample rate of the DAC.
- **Output bit depth:** sets the bit depth of the DAC. higher values provide more resolution.
- **Maximum volume:** sets the max value in the volume column.
- **Stereo:** when enabled, you may use panning effects.
- **Interpolation:** "softens" samples played back at lower rates.

# Dave

---

this is the sound chip used in the Enterprise 128 home computer of the '80s, which competed against other home computers in Europe such as the ZX Spectrum and Amstrad CPC.

Dave is very similar to POKEY in many aspects. it has most of the signature Atari sounds and POKEY-style high-pass filter.

it has 4 channels, of which 3 generate LFSR-based sounds and the last one is a noise channel which has five lengths and either runs at a fixed frequency, or steals the frequency of another channel.

these channels have ring modulation and the aforementioned high-pass filter capabilities. the noise one also has a pseudo-low-pass filter.

the pitch and volume resolutions are much greater than that of POKEY, with 4096 pitches and 64 volume levels.

it also has stereo output.

on top of that, there's a DAC mode which may be enabled for each side of the stereo output. this mode overrides sound generation output.

## effects

---

- 10xx: **set waveform or noise length.**
- the following waveforms apply in the first three channels:
  - 0: square
  - 1: bass
  - 2: buzz
  - 3: reed
  - 4: noise
- if placed in the noise channel, x is a value from 0 to 3.
- 11xx: **set noise frequency source.**
- 0: fixed frequency (~62.5KHz)
- 1: channel 1
- 2: channel 2
- 3: channel 3
- 12xx: **toggle high-pass with the next channel.**
- 13xx: **toggle ring modulation with the channel that is located two channels ahead of this one.**
- in the case of channel 1, it modulates with channel 3. channel 2 modulates with channel 4 and so on.
- 14xx: **toggle "swap counters" mode.**
- only in noise channel.
- when enabled, the noise length is even shorter and has no effect.
- 15xx: **toggle low-pass with channel 2.**
- only in noise channel.
- 16xx: **set global clock divider.**
- 0: divide by 2.
- 1: divide by 3.

# info

---

this chip uses the [Dave<sup>132</sup>](#) instrument editor.

when two channels are joined due to high-pass filter, the channel bar will show `high` on a bracket tying them together.

when two channels are joined due to low-pass filter, the channel bar will show `low` on a bracket tying them together.

when two channels are joined for ring modulation, the channel bar will show `ring` on a bracket tying them together.

# Ensoniq ES5506 (OTTO)

---

sample-based synthesis chip used in a bunch of Taito arcade machines and PC sound cards like Soundscape Elite. a variant of it was the heart of the well-known Gravis Ultrasound.

it has a whopping 32 channels of 16-bit PCM and:

- real time digital filters
- linear interpolation
- loop start and stop positions for each voice (bidirectional and reverse looping)
- internal volume multiplication and stereo panning
- hardware support for short envelopes

## effects

---

- 11xx: **set filter mode.** values are 0 through 3.
- 120x: **set pause (bit 0).** pauses the sample until the bit is unset; it will then resume where it left off.
- 14xx: **set filter coefficient K1 low byte.**
- 15xx: **set filter coefficient K1 high byte.**
- 16xx: **set filter coefficient K2 low byte.**
- 17xx: **set filter coefficient K2 high byte.**
- 18xx: **set filter coefficient K1 slide up.**
- 19xx: **set filter coefficient K1 slide down.**
- 1Axx: **set filter coefficient K2 slide up.**
- 1Bxx: **set filter coefficient K2 slide down.**
- 20xx: **set envelope count.**
- 22xx: **set envelope left volume ramp.**
- 23xx: **set envelope right volume ramp.**
- 24xx: **set envelope filter coefficient K1 ramp.**
- 25xx: **set envelope filter coefficient K1 ramp (slower).**
- 26xx: **set envelope filter coefficient K2 ramp.**
- 27xx: **set envelope filter coefficient K2 ramp (slower).**
- 3xxx: **set coarse filter coefficient K1.**
- 4xxx: **set coarse filter coefficient K2.**
- 81xx: **set panning (left channel).**
- 82xx: **set panning (right channel).**
- 88xx: **set panning (rear channels).**
- 89xx: **set panning (rear left channel).**
- 8Axx: **set panning (rear right channel).**
- 9xxx: **set sample offset.**
- resets sample position to  $xxx * 0x100$ .
- DFxx: **set sample playback direction.**

## info

---

this chip uses the [ES5506<sub>133</sub>](#) instrument editor.

# chip config

---

the following options are available in the Chip Manager window:

- **Initial channel limit:** sets how many channels are available for use. if reduced, output rate increases.
- **Volume scale:** allows you to lower the overall volume to prevent clipping/distortion when using too many channels.
- **Amiga channel volumes:** makes volume linear, from 0 to 64 (40 in hex). used in S3M, XM and IT import.
- **Amiga-like pitch:** pretends to be an Amiga, with periodic slides. used in S3M, XM and IT import (the latter two when linear slides are disabled).
- only effective when pitch linearity is None.

# ESS ESFM

---

an FM synthesizer core included in a series of sound card chipsets made by ESS, which were mildly popular in the DOS days during the mid-late 90s.

at a cursory glance, it looks like just an [OPL3 clone<sup>268</sup>](#), but hidden under a veil of mystery is its exclusive "native mode", revealing an impressive superset of features, including 4-operator support on all 18 channels, semi-modular operator routing, per-operator pitch control, and even a few unique features.

for a long time, not much was known about the inner workings of ESFM's native mode, since ESS did not release any documentation to developers on how to use it. this has thankfully changed in recent years thanks to reverse-engineering efforts from the community.

thanks to ESS's decision to not release any documentation to developers and lock down usage of native mode behind a couple of General MIDI drivers shipping with rather lackluster patch sets, ESFM's native mode was unfortunately not very well used over its original lifespan.

## effects

---

- 10xy: **set AM depth.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is either 0 (1dB, shallow) or 1 (4.8dB, deep).
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value..
- 17xy: **set vibrato depth.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is either 0 (normal) or 1 (double).
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 20xy: **set panning of operator 1.**
- x determines whether to output on left.
- y determines whether to output on right.
- 21xy: **set panning of operator 2.**
- x determines whether to output on left.
- y determines whether to output on right.
- 22xy: **set panning of operator 3.**
- x determines whether to output on left.
- y determines whether to output on right.
- 23xy: **set panning of operator 4.**



- x determines whether to output on left.
- y determines whether to output on right.
- 24xy: **set output level of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 25xy: **set modulation input level of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 26xy: **set envelope delay of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 27xx: **set operator 4 noise mode.**
- 0: noise off
- 1: square + noise
- 2: ring mod from operator 3 + noise
- 3: ring mod from operator 3 + double pitch modulation input
  - note: emulation issues. subject to change!
- 2Axy: **set waveform of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 2Exx: **enable envelope hard reset.**
- 2Fxy: **set fixed frequency block (octave).**
- x is the operator from 1 to 4.
- y is the block/octave from 0 to 7.
- 3xyy: **set fixed frequency f-num.**
- x contains operator number and high bits of f-num may be any of the following:
  - 0 to 3 for operator 1
  - 4 to 7 for operator 2
  - 8 to B for operator 3
  - C to F for operator 4
- y are the lower bits of f-num.
- 40xx: **set operator 1 detune.**
- 41xx: **set operator 2 detune.**
- 42xx: **set operator 3 detune.**
- 43xx: **set operator 4 detune.**
- 50xy: **set AM of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 53xy: **set VIB of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y determines whether VIB is on.
- 54xy: **set KSL of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y is the value.
- 55xy: **set SUS of operator.**

- x is the operator from 1 to 4. a value of 0 means "all operators".
- y determines whether SUS is on.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxy: **set KSR of operator.**
- x is the operator from 1 to 4. a value of 0 means "all operators".
- y determines whether KSR is on.

## info

---

this chip uses the [FM \(ESFM\)](#)<sub>135</sub> instrument editor.

# Famicom Disk System

---

the Famicom Disk System is an expansion device for the Famicom (known as NES outside Japan), a popular console from the '80s.

as its name implies, it allowed people to play games on specialized floppy disks that could be re-written on vending machines, therefore reducing the cost of ownership and manufacturing.

it also offers an additional 6-bit, 64-byte wavetable sound channel with (somewhat limited) FM capabilities, which is what Furnace supports.

## effects

---

- 10xx: **change wave.**
- 11xx: **set modulation depth.**
- 12xy: **set modulation speed high byte and toggle on/off.**
- x is the toggle. a value of 1 turns on the modulator.
- y is the speed.
- 13xx: **set modulation speed low byte.**
- 14xx: **set modulator position.**
- 15xx: **set modulator wave.**
- xx points to a wavetable. it should (preferably) have a height of 7 with the values mapping to:
  - 0: +0
  - 1: +1
  - 2: +2
  - 3: +3
  - 4: reset
  - 5: -3
  - 6: -2
  - 7: -1
- **do not use this effect.** it only exists for compatibility reasons

## info

---

this chip uses the [FDS<sub>134</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# Irem GA20

---

it is a 4 channel PCM sound source used by Irem in their arcades in late 1980s and early 1990s, of ten paired with [Yamaha YM2151](#)<sub>328</sub>.

the sound chip itself is rather unremarkable, having 8-bit volume and pitch control and no stereo panning...

## effects

---

none

let's be honest. Furnace has too many chips and a great portion of them are sample chips that do the same task: playing back samples.

## info

---

this chip uses the [GA20](#)<sub>154</sub> instrument type.

# Game Boy

---

the Game Boy is one of the most successful portable game systems ever made.

it has stereo sound, two pulse channels, a wave channel and a noise channel.

## effects

---

- 10xx: **change wave.**
- 11xx: **set noise length.**
- 0: long
- 1: short
- 12xx: **set duty cycle.**
- 0: 12.5%
- 1: 25%
- 2: 50%
- 3: 75%
- 13xy: **setup sweep.** pulse 1 only.
- x is the time.
- y is the shift.
- set to 0 to disable it.
- 14xx: **set sweep direction.** 0 is up and 1 is down.

## info

---

this chip uses the [Game Boy<sub>155</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Disable anti-click:** waveform switching requires a phase reset, which may cause clicks. Furnace uses a wave-position predicting algorithm to minimize these clicks. enable this option to disable it.
- **Chip revision:** sets the chip model to use. most of these lack audible difference, but Game Boy Advance fixed the wave channel's inversion.
- **Wave channel orientation:** allows you to set how is wave data written.
- in Game Boy:
  - Exact data: wave data is written as-is. it will appear inverted in the output.
  - Exact output: wave data is inverted so it appears correctly in the output.
- in Game Boy Advance:
  - Normal: wave data is written as-is.
  - Inverted: guess!
- **Pretty please:** only for compatibility with Synchronize.dmf. do not use.

## links

---

- [Gameboy sound hardware](https://gbdev.gg8.se/wiki/articles/Gameboy_sound_hardware) ([https://gbdev.gg8.se/wiki/articles/Gameboy\\_sound\\_hardware](https://gbdev.gg8.se/wiki/articles/Gameboy_sound_hardware))
- detailed technical information
- [GameBoy Sound Table](http://www.devrs.com/gb/files/sndtab.html) (<http://www.devrs.com/gb/files/sndtab.html>) - note frequency table

# Game Boy Advance

---

a portable video game console from Nintendo, succeeding the Game Boy.

it adds two stereo sample audio channels which can be used directly ("DMA", left/right) or used in a software mixing driver (most games do this) in order to have multiple voices.

## effects

---

- 10xx: **change wave.**

### Game Boy Advance (MinMod)

---

this is the software mixing driver available in Furnace. it is written by Natt Akuma. it features echo and up to 16 voices.

- 10xx: **change wave.**
- 11xy: **configure echo.**
- this effect is kinda odd. here's how to use it:
  - create an empty instrument and put a very high note of it in channel 1.
  - put 110x in the effect column.
  - set volume column to set feedback.
  - don't use the channel for anything else.
- 12xy: **toggle invert.**
- x left channel.
- y right channel.

## info

---

this chip uses the [GBA DMA<sub>157</sub>](#) and [GBA MinMod<sub>158</sub>](#) instrument editors.

## chip config

---

the following options are available in the Chip Manager window:

- **DAC bit depth:** sets the bit depth.

these options are available when using MinMod:

- **Volume scale:** scale volumes to prevent clipping/distortion.
- **Mix buffers:** sets how many mix buffers will be stored in memory. higher values result in longer echo.
- **Channel limit:** sets the number of channels that will be available. higher values use more CPU.
- **Sample rate:** sets the mixing rate. higher values use more CPU.

# Konami K007232

---

a 2-channel PCM sound chip from Konami which was used in some of their 1986-1990 arcade boards.

its sample format is unique; the topmost bit is the end marker, and the low 7 bits are used for generating sound (unsigned format).

it has 7 bit digital output per each channel and no volume register on chip, so it needs external logic to control channel volume.

## effects

---

nothing.

yeah.

## info

---

this chip uses the [K007232<sub>159</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Stereo**: enable use of panning effects.



# Konami K053260

---

this chip is a sample-based chip that featured in a number of Konami arcade games, notably *Sunset Riders* and *Teenage Mutant Ninja Turtles: Turtles in Time*. it has four channels of audio, 12-bit pitch resolution and stereo output, and can access up to 2MB of samples in 8-bit PCM or 4-bit AD PCM formats.

## effects

---

- DFxx: **set sample playback direction.**
- 0 is normal.
- 1 is reverse.

## info

---

this chip uses the [K053260<sub>160</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# Atari Lynx/MIKEY

---

the Atari Lynx is a 16 bit handheld console developed by Atari Corporation, and initially released in September of 1989, with the worldwide release being in 1990.

while it was an incredible handheld for the time (and a lot more powerful than Game Boy), it unfortunately meant nothing in the end due to Lynx being a market failure, and ending up as one of the things that contributed to the downfall of Atari.

although the Lynx is still getting (rather impressive) homebrew developed for it, that does not mean the Lynx is a popular system at all.

but hey, Furnace supports it, so...

the Atari Lynx has a 6502-based CPU with a sound part (this chip is known as MIKEY). it has the following sound capabilities:

- 4 channels of LFSR-based sound, which can be modulated with different frequencies ( $\div 0$ ,  $\div 1$ ,  $\div 2$ ,  $\div 3$ ,  $\div 4$ ,  $\div 5$ ,  $\div 7$ ,  $\div 10$ , and  $\div 11$ ) to create square waves and wavetable-like results.
- likewise, when a lot of the modulators are activated, this can provide a "pseudo-white noise"-like effect, which can be useful for drums and sound effects.
- soft stereo panning capabilities via the 08xx effect command.
- four 8-bit DACs (Digital to Analog Converter), one for each voice. this allows for sample playback (at the cost of CPU time and memory).
- a variety of pitches to choose from, and they go from 32Hz to "above the range of human hearing", according to Atari.

## effects

---

- 3xxx: **load LFSR**. this is a bitmask with values ranging from 000 to FFF.
- for it to work, duty macro in instrument editor must be set to some value. without it LFSR will not be fed with any bits.

## info

---

this chip uses the [Atari Lynx<sup>161</sup>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Consistent frequency across all duties**: uses an algorithm to calculate frequency depending on duty.
- make sure you reset phase on each note. otherwise sudden changes in the LFSR may make this option less effective.

# Nintendo MMC5

---

a mapper chip which made NES cartridges exceeding 1MB possible.

it has two pulse channels which are very similar to the ones found in the NES, but lacking the sweep unit.

additionally, it offers an 8-bit DAC which can be used to play samples. only one game is known to use it, though.

## effects

---

- 12xx: **set duty cycle or noise mode of channel.**
- may be 0 through 3 for the pulse channels.

## info

---

this chip uses the [NES<sub>170</sub>](#) and [Generic Sample<sub>183</sub>](#) instrument editors.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# OKI MSM5232

---

a rather primitive (but powerful) sound generator chip used in some arcade boards and even synthesizers (like Korg Poly-800).

it has 8 channels in 2 groups (of 4 channels each). each group can be configured with an envelope and the ability to produce 2', 4', 8' and/or 16' square/noise overtones on 8 outputs (four (2', 4', 8' and 16') per group).

however, this chip cannot entirely produce sound alone. it has to be paired with either an external envelope generator or capacitors to assist the internal envelope generator.

it also has no fine tune whatsoever. the frequency resolution is exactly a semitone.

Furnace implements this chip in a way that allows the following features:

- internal (capacitors) or external (volume macro) envelope modes per group
- the ability to change the volume of each output (this can be used to generate saw waves if you set each part/overtone's volume to exactly half of the previous one)
- global fine tune
- global vibrato (some arcade boards played with the clock input to simulate vibrato)

## effects

---

- 10xy: **set group control**.
- x sets sustain mode.
- y is a 4-bit mask which toggles overtones.
- 11xx: **set noise mode**.
- 12xx: **set group attack**. range is 0 to 5.
- only in internal (capacitor-based) envelope mode.
- 13xx: **set group decay**. range is 0 to 11.
- only in internal (capacitor-based) envelope mode.

## info

---

this chip uses the [MSM5232<sub>163</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Detune**: sets detune in 128ths of a semitone.
- **Capacitor values**: allows you to change the capacitance of the 8 capacitors connected to the chip. longer values increase envelope length.
- **Initial part volume**: sets the initial volume of each "part" (octave) in a group.
- **Envelope mode**: sets whether to use an external volume input (in this case, macros) or the built-in envelope generator.
- **Global vibrato**: nudges the clock to simulate vibrato.

# OKI MSM6258

---

a single-channel ADPCM sound source developed by OKI. it allows max sample rate of 15.6 KHz... with no variable pitch. most prominent use of this chip was Sharp X68000 computer, where it was paired with Yamaha YM2151.

Furnace's implementation is MSM6258V, a CPU driven variant that is unlimited by amount of sample data, being able to be fed from the system's RAM.

## effects

---

- 20xx: **set frequency divider (0 to 2).**
- 0: /512
- 1: /768
- 2: /1024
- 21xx: **select clock rate.**
- 0: full
- 1: half

## chip config

---

MSM6258 is an extremely basic ADPCM sound codec. it has no variable frequency rate; it depends on clock rate of a chip itself. Furnace supports the following rates:

CLOCK RATE	SAMPLING RATE
4 MHz	7812 Hz
4.096 MHz	8000 Hz
8 MHz	15625 Hz
8.192 MHz	16000 Hz

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

## info

---

this chip uses the [MSM6258<sub>164</sub>](#) instrument editor.

# OKI MSM6295

---

an upgrade from 6258 - it provides 4 ADPCM channels, at max 32 KHz (still no variable pitch though). between late '80s and late '90s, it was one of the most common, if not *the* most common soundchip used in arcade machines (Capcom, Toaplan, Kaneko, Atari, Tecmo, the list can go on and on...). without bankswitching, the chip supports 256kB of sample RAM and can hold up to 127 samples at once.

## effects

---

- 20xx: **set chip output rate.**
- 0: /132
- 1: /165

## info

---

this chip uses the [MSM6295<sub>165</sub>](#) instrument editor.

## chip clock rates

---

like MSM6258, MSM6295 is an extremely basic ADPCM sound codec. it has no variable frequency rate, it depends on clock rate of a chip itself. Furnace supports following rates:

CLOCK RATE	SAMPLING RATE
1 MHz	7576 Hz
1.02 MHz	7727 Hz
1.056 MHz	8000 Hz
1.193 MHz	9038 Hz
0.89 MHz	6742 Hz
0.875 MHz	6629 Hz
0.9375 MHz	7102 Hz
1.5 MHz	11364 Hz
1.79 MHz	13561 Hz
2 MHz	15152Hz
2.112 MHz	16000 Hz
3 MHz	22728 Hz
3.58 MHz	27122 Hz
4 MHz	30304 Hz
4.224 MHz	32000 Hz
### chip clock divisor	

MSM6295 clock rate could be divided by 132 (resulting sample rates above), or by 165.  
to get a clock rate using divisor of 165, formula is clock rate (in Hz) / 165.  
example: 1 MHz MSM6295 in 165 divisor mode results in output rate of 6060 Hz.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Output rate:** sets the chip clock divider.
- **Bankswitched:** add an NMK112 chip to increase sample memory.

# Namco 163 (also called N163, Namco C163, Namco 106 [sic], Namco 160 or Namco 129)

---

this is one of Namco's NES mappers, with up to 8 wavetable channels.

it has 256 nibbles (128 bytes) of internal RAM which is shared between channel state and waves.

wavetables are variable in size and may be allocated anywhere in RAM. at least 128 nibbles (64 bytes) can be dedicated to waves, with more available if not all channels are used - waveform RAM area becomes smaller as more channels are activated, since channel registers consume 8 bytes for each channel.

Namco 163 uses time-division multiplexing (TDM) for its output. this means that only one channel is output per sample (like OPLL and OPN2). therefore, its sound quality gets worse as more channels are activated.

## waveform load position versus waveform position

---

in Furnace, waveform **load** position/length is different from the waveform position/length.

when placing a note, the load pos/len and the pos/len are set to the values specified in the instrument.

waveforms are loaded in the region set by the **load** pos/len, which you can change using effects 15xx and 16xx as described below.

the region that will play is set by the waveform pos/len, which you can alter using effects 11xx and 12xx.

the waveform pos/len macros only change the pos/len, and not the **load** one.

if the waveform changes (e.g. ins change, wave macro or wave synth), or the **load** pos/len changes, the wave is written to memory.

## effects

---

- 10xx: **set waveform for playback.**
- 11xx: **set waveform position in RAM for playback.**
- 12xx: **set waveform length in RAM for playback.**
- x goes from 04 to FC in steps of 4.



- 15xx: **set waveform load position.**
- 16xx: **set waveform load length.**
- x goes from 04 to FC in steps of 4.
- 180x: **set channel limit.**
- range of x is 0 to 7. 1 is added to get results of 1 through 8.
- 20xx: **load a waveform to RAM.**
- x is the waveform.
- the length is determined by the wave's width (it will be snapped to a multiple of 4 if it isn't).
- make sure to use 21xx first!
- 21xx: **set position for 20xx.**

## info

---

this chip uses the [Namco 163<sub>168</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Initial channel limit:** sets the number of channels that will be active. higher values reduce volume and make TDM artifacts more noticeable.
- **Disable hissing:** remove TDM artifacts by mixing. sacrifices some accuracy!
- **Scale frequency to wave length:** automatically adjusts note frequency to account for differing waveform lengths.
- if disabled, note frequencies ignore waveveform length. this is how FamiTracker behaves.

# Namco WSG / Namco C15 / Namco C30

---

a family of wavetable synth sound chips used by Namco in their arcade machines (Pac-Man and later). waveforms are 4-bit, with 32-byte sample length.

everything starts with Namco WSG, which is a simple 3-channel wavetable with no extra frills. C15 is a much more advanced sound source with 8 channels, and C30 adds stereo output and noise mode.

## effects

---

- 10xx: **change waveform.**
- 11xx: **toggle noise mode.** *warning:* only on C30.

## info

---

this chip uses the [Namco WSG<sub>208</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Waveform storage mode:** selects whether RAM or ROM is connected to the chip.
- RAM: connects RAM for unlimited waves.
- ROM: connects ROM. authentic to hardware, but only the first 8 waves are loaded.

Namco C30 does not have the aforementioned option as it always uses RAM for waveforms. instead, it has this option:

- **Compatible noise frequencies:** for compatibility with old Furnace.

# Nintendo DS

---

this portable video game console succeeded the Game Boy Advance.

it has 16 channels of sampled sound, supporting 8-bit PCM, 16-bit PCM and IMA ADPCM.

additionally, the last 8 channels may be put in "PSG mode", featuring 6 channels of pulse with 8 duty cycles and 2 noise channels.

## effects

---

- 12xy: **set duty cycle**.
- 0 to 7.
- only works in PSG channels.
- 1Fxx: **set global volume**.

## info

---

this chip uses the [Nintendo DS<sub>169</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Model**: set the amount of memory available for samples.

# NES

---

the console from Nintendo that plays Super Mario Bros. and helped revive the agonizing video game market in the US during mid-80s.

also known as Family Computer (Famicom), especially in Japan.

the console is powered by the Ricoh 2A03, a CPU with sound generator built-in. it has five channels: first two channels play pulse wave with three different duty cycles, third is a fixed-volume triangle channel, fourth is a noise channel (can work in both pseudo-random and periodic modes) and fifth is a (D)PCM sample channel.

## effects

---

- 11xx: **write to delta modulation counter.** range is 00 to 7F.
  - this may be used to attenuate the triangle and noise channels; at 7F, they will be at about 57% volume.
  - will not work if a sample is playing.
- 12xx: **set duty cycle or noise mode of channel.**
  - may be 0 to 3 for the pulse channels:
    - 0: 12.5%
    - 1: 25%
    - 2: 50%
    - 3: 75%
  - may be 0 or 1 for the noise channel:
    - 0: long (15-bit LFSR, 32767-step)
    - 1: short (9-bit LFSR, 93-step)
    - short noise may sound very different depending on its initial LFSR state. more info can be found at [NESdev](https://forums.nesdev.org/viewtopic.php?t=11535) (<https://forums.nesdev.org/viewtopic.php?t=11535>) .
- 13xy: **setup sweep up.**
  - x is the time.
  - y is the shift.
  - set to 0 to disable it.
- 14xy: **setup sweep down.**
  - x is the time.
  - y is the shift.
  - set to 0 to disable it.
- 15xx: **set envelope mode.**
  - 0: envelope + length counter. volume represents envelope duration.
  - 1: length counter. volume represents output volume.
  - 2: looping envelope. volume represents envelope duration.
  - 3: constant volume. default value. volume represents output volume.
  - pulse and noise channels only.
  - you may need to apply a phase reset (using the macro) to make the envelope effective.
- 16xx: **set length counter.**
  - see table below for possible values.
  - this will trigger phase reset.
- 17xx: **set frame counter mode.**

- 0: 4-step.
  - NTSC: 120Hz sweeps and lengths; 240Hz envelope.
  - PAL: 100Hz sweeps and lengths; 200Hz envelope.
  - Dendy: 118.9Hz sweeps and lengths; 237.8Hz envelope.
- 1: 5-step.
  - NTSC: 96Hz sweeps and lengths; 192Hz envelope.
  - PAL: 80Hz sweeps and lengths; 160Hz envelope.
  - Dendy: 95.1Hz sweeps and lengths; 190.2Hz envelope.
- 18xx: **set PCM channel mode.**
- 00: PCM (software).
- 01: DPCM (hardware).
- when in DPCM mode, samples will sound muffled (due to its nature) and available pitches are limited. see "DPCM samples" below.
- 19xx: **set triangle linear counter.**
- 00 to 7F set the counter.
- 80 and higher halt it.
- 20xx: **set DPCM frequency.**
- only works in DPCM mode.
- see table below for possible values.

## info

---

this chip uses the [NES<sub>170</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **DPCM channel mode:** allows you to set which mode to use for the DPCM channel.
- DPCM: the default mode, playing 1-bit DPCM samples as supported by the hardware.
- PCM: this mode provides crisper 7-bit samples by writing to the delta counter directly. uses a lot of CPU time in console.

## DPCM samples

---

due to hardware limitations, a loop in a DPCM sample must start on a multiple of 512 samples (512, 1024, 1536...) and have a length that is a multiple of 128 plus 8 samples (136, 264, 392...)

NES DPCM only has 16 preset sample rates, shown in a table below. for help adapting samples to work with this, see the [limited samples guide](#)<sup>396</sup>.

## short noise frequencies (NTSC)

---

NOTE	ARPEGGIO	FUNDAMENTAL	MIDI NOTE	PITCH
C-0	@0	4.7 Hz	-9.47	d_1 + 53¢
C#0	@1	9.5 Hz	2.53	D-0 + 53¢
D-0	@2	18.9 Hz	14.55	D-1 + 55¢

NOTE	ARPEGGIO	FUNDAMENTAL	MIDI NOTE	PITCH
D#0	@3	25.3 Hz	19.53	G - 1 + 53¢
E - 0	@4	37.9 Hz	26.55	D - 2 + 55¢
F - 0	@5	50.6 Hz	31.57	G - 2 + 57¢
F#0	@6	75.8 Hz	38.55	D - 3 + 55¢
G - 0	@7	95.3 Hz	42.51	F#3 + 51¢
G#0	@8	120.3 Hz	46.55	A#3 + 55¢
A - 0	@9	150.4 Hz	50.41	D - 4 + 41¢
A#0	@10	200.5 Hz	55.39	G - 4 + 39¢
B - 0	@11	300.7 Hz	62.41	D - 5 + 41¢
C - 1	@12	601.4 Hz	74.41	D - 6 + 41¢
C#1	@13	1202.8 Hz	86.41	D - 7 + 41¢
D - 1	@14	2405.6 Hz	98.41	D - 8 + 41¢
D#1	@15	4811.2 Hz	110.41	D - 9 + 41¢

reference: [NESdev](https://www.nesdev.org/wiki/APU_Noise) ([https://www.nesdev.org/wiki/APU\\_Noise](https://www.nesdev.org/wiki/APU_Noise))

## length counter table

VALUE	RAW	NTSC	PAL	DENDY	NTSC 5-STEP	PAL 5-STEP	DENDY 5-STEP
03	2	17ms	20ms	17ms	21ms	25ms	21ms
05	4	33ms	40ms	34ms	42ms	50ms	42ms
07	6	50ms	60ms	50ms	63ms	75ms	63ms
09	8	67ms	80ms	67ms	83ms	100ms	84ms
00	10	83ms	100ms	84ms	104ms	125ms	105ms
0B	10	83ms	100ms	84ms	104ms	125ms	105ms
0D	12	100ms	120ms	101ms	125ms	150ms	126ms
10	12	100ms	120ms	101ms	125ms	150ms	126ms
0C	14	117ms	140ms	118ms	146ms	175ms	147ms
0F	14	117ms	140ms	118ms	145ms	175ms	147ms
1C	16	133ms	160ms	135ms	167ms	200ms	168ms
11	16	133ms	160ms	135ms	167ms	200ms	168ms
13	18	150ms	180ms	151ms	188ms	225ms	189ms
02	20	166ms	200ms	168ms	208ms	250ms	210ms
15	20	167ms	200ms	168ms	208ms	250ms	210ms
17	22	183ms	220ms	185ms	229ms	275ms	231ms
12	24	200ms	240ms	202ms	250ms	300ms	252ms
19	24	200ms	240ms	202ms	250ms	300ms	252ms
0E	26	217ms	260ms	219ms	271ms	325ms	273ms

VALUE	RAW	NTSC	PAL	DENDY	NTSC 5-STEP	PAL 5-STEP	DENDY 5-STEP
1B	26	217ms	260ms	219ms	271ms	325ms	273ms
1D	28	233ms	280ms	235ms	292ms	350ms	294ms
1F	30	250ms	300ms	252ms	313ms	375ms	315ms
1E	32	267ms	320ms	269ms	333ms	400ms	336ms
04	40	333ms	400ms	336ms	417ms	500ms	421ms
14	48	400ms	480ms	404ms	500ms	600ms	505ms
0A	60	500ms	600ms	505ms	625ms	750ms	631ms
1A	72	600ms	720ms	606ms	750ms	900ms	757ms
06	80	667ms	800ms	673ms	833ms	1.0s	841ms
16	96	800ms	960ms	807ms	1.0s	1.2s	1.0s
08	160	1.3s	1.6s	1.3s	1.7s	2.0s	1.7s
18	192	1.6s	1.9s	1.6s	2.0s	2.4s	2.0s
01	254	2.1s	2.5s	2.1s	2.6s	3.2s	2.7s

reference: [NESdev](https://www.nesdev.org/wiki/APU_Length_Counter) ([https://www.nesdev.org/wiki/APU\\_Length\\_Counter](https://www.nesdev.org/wiki/APU_Length_Counter))

## DPCM frequency table

VALUE	TRACKER	NTSC FREQ	NTSC PITCH	PAL FREQ	PAL PITCH
00	C-3	4181.7Hz	C-8 - 2¢	4177.4Hz	C-8 - 4¢
01	D-3	4709.9Hz	D-8 + 4¢	4696.6Hz	D-8 - 1¢
02	E-3	5264.0Hz	E-8 - 3¢	5261.4Hz	E-8 - 4¢
03	F-3	5593.0Hz	F-8 + 2¢	5579.2Hz	F-8 - 3¢
04	G-3	6258.0Hz	G-8 - 4¢	6023.9Hz	G-8 - 70¢
05	A-3	7046.4Hz	A-8 + 2¢	7044.9Hz	A-8 + 1¢
06	B-3	7919.4Hz	B-8 + 4¢	7917.2Hz	B-8 + 3¢
07	C-4	8363.4Hz	C-9 - 2¢	8397.0Hz	C-9 + 5¢
08	D-4	9419.9Hz	D-9 + 4¢	9446.6Hz	D-9 + 9¢
09	F-4	11186.1Hz	F-9 + 2¢	11233.8Hz	F-9 + 9¢
0A	G-4	12604.0Hz	G-9 + 8¢	12595.5Hz	G-9 + 7¢
0B	A-4	13982.6Hz	A-9 - 12¢	14089.9Hz	A-9 + 1¢
0C	C-5	16884.6Hz	C-10 + 15¢	16965.4Hz	C-10 + 23¢
0D	E-5	21306.8Hz	E-10 + 17¢	21315.5Hz	E-10 + 18¢
0E	G-5	24858.0Hz	G-10 - 16¢	25191.0Hz	G-10 + 7¢
0F	C-6	33143.9Hz	C-11 - 18¢	33252.1Hz	C-11 - 12¢

reference: [NESdev](https://www.nesdev.org/wiki/APU_DMC#Pitch_table) ([https://www.nesdev.org/wiki/APU\\_DMC#Pitch\\_table](https://www.nesdev.org/wiki/APU_DMC#Pitch_table))

# Yamaha OPL

---

a series of FM sound chips which were very popular in DOS land. it was so popular that even Yamaha made a logo for it!

essentially a downgraded version of Yamaha's other FM chips, with only 2 operators per channel. however, it also had a [drums mode](#)<sup>271</sup>, and later chips in the series added more waveforms (than just the typical sine) and even a 4-operator mode.

the original OPL (Yamaha YM3526) was present as an expansion for the Commodore 64 and MSX computers (erm, a variant of it). it only had 9 two-operator channels and drums mode.

Y8950 is essentially the same chip, but with one additional channel for ADPCM sample playback, behaving in a similar way as the one found in YM2608.

its successor, the OPL2 (Yamaha YM3812), added 3 more waveforms and was one of the more popular chips because it was present on the AdLib card for PC.

later Creative would borrow the chip to make the Sound Blaster, and totally destroyed AdLib's dominance.

the OPL3 (Yamaha YMF262) added 9 more channels, 4 more waveforms, rudimentary 4-operator mode (pairing up to 12 channels to make up to six 4-operator channels), quadraphonic output and some other things.

the OPL4 (Yamaha YMF278) retains the FM block from OPL3, but adds 24 sample channels on top! samples are 44100 Hz, can be between 8, 12 or 16-bit, stereo (with 16 different levels) and can read PCM data from ROM or SRAM (up to 4 MB).

afterwards everyone moved to Windows and software mixed PCM streaming...

## effects

---

- 10xx: **set AM depth.** the following values are accepted:
  - 0: 1dB (shallow)
  - 1: 4.8dB (deep)
- this effect applies to all channels.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- only in 4-op mode (OPL3 and OPL4).
- 15xx: **set operator 4 level.**
- only in 4-op mode (OPL3 and OPL4).
- 16xy: **set multiplier of operator.**
- x is the operator (1-4; last 2 operators only in 4-op mode).
- y is the new MULT value..
- 17xx: **set vibrato depth.**
- 0: normal
- 1: double



- this effect applies to all channels.
- 18xx: **toggle drums mode.**
- 0 disables it and 1 enables it.
- only in drums chip.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- only in 4-op mode (OPL3).
- 1Dxx: **set attack of operator 4.**
- only in 4-op mode (OPL3).
- 2Axy: **set waveform of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y is the value.
- only in OPL2 or higher.
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y is the value.
- 53xy: **set VIB of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y determines whether VIB is on.
- 54xy: **set KSL of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y is the value.
- 55xy: **set SUS of operator.**
- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y determines whether SUS is on.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- only in 4-op mode (OPL3).
- 5Axx: **set DR of operator 4.**
- only in 4-op mode (OPL3).
- 5Bxy: **set KSR of operator.**

- x is the operator from 1 to 4; the last 2 operators only work in 4-op mode. a value of 0 means "all operators".
- y determines whether KSR is on.

## info

---

these chips use the [FM \(OPL\)](#)<sub>139</sub> instrument editor.

Y8950 ADPCM uses [Generic Sample](#)<sub>183</sub> and [ADPCM-B](#)<sub>123</sub> instrument editors.

OPL4 PCM uses the [MultiPCM](#)<sub>166</sub> instrument editor.

when two channels are joined for 4-op mode, the channel bar will show 40P on a bracket tying them together.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

additionally, in OPL3:

- **Chip type:** sets the chip model. OPL3-L uses resampling.
- **Compatible panning:** for compatibility with old Furnace.

additionally, in OPL4:

- **RAM Size:** sets the RAM size for sample storage.

# Yamaha YM2413/OPLL

---

the YM2413, otherwise known as OPLL, is a cost-reduced FM synthesis sound chip, based on the Yamaha YM3812 (OPL2).

OPLL also spawned a few derivative chips, the best known of these is:

- the famous Konami VRC7. used in the Japan-only video game Lagrange Point, it was **another** cost reduction on top of the OPLL! this time just 6 channels...
- Yamaha YM2423, same chip as YM2413, just a different patch set...
- Yamaha YMF281, ditto.....

## technical specifications

---

the YM2413 is equipped with the following features:

- 9 channels of 2 operator FM synthesis
- a drum/percussion mode, replacing the last 3 voices with 5 rhythm channels, with drum mode tones hard-defined in the chip itself, like FM instruments. only pitch might be altered.
- drum mode works like following: FM channel 7 is for Kick Drum, which is a normal FM channel but routed through mixer twice for 2x volume, like all drum sounds. FM channel 8 splits to Snare, Drum, and Hi-Hat. Snare Drum is the carrier and it works with a special 1 bit noise generator combined with a square wave, all possible by overriding phase-generator with some different synthesis method. Hi-Hat is the modulator and it works with the noise generator and also the special synthesis. channel 9 splits to Top-Cymbal and Tom-Tom, Top-Cymbal is the carrier and only has the special synthesis, while Tom-Tom is basically a 1op wave.
- special synthesis mentioned already is: 5 square waves are gathered from 4x, 64x and 128x the pitch of channel 8 and 16x and 64x the pitch of channel 9 and they go through a process where 2 HH bits OR'd together, then 1 HH and 1 TC bit OR'd, then the two TC bits OR'd together, and those 3 results get XOR'd.
- **1 user-definable patch (this patch can be changed throughout the course of the song)**
- **15 pre-defined patches which can all be used at the same time**
- support for ADSR on both the modulator and the carrier
- sine and half-sine based FM synthesis
- 9 octave note control
- 4096 different frequencies for channels
- 16 unique volume levels (NOTE: volume 0 is NOT silent.)
- modulator and carrier key scaling
- built-in hardware vibrato support

## effects

---

- 10xx: **change patch.**
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 16xy: **set multiplier of operator.**
- x is the operator, either 1 or 2.

- y is the new MULT value..
- 18xx: **toggle drums mode.**
- 0 disables it and 1 enables it.
- only in drums mode.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 50xy: **set AM of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y is the value.
- 53xy: **set VIB of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y determines whether VIB is on.
- 54xy: **set KSL of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y is the value.
- 55xy: **set EGT of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y determines whether EGT is on.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 5Bxy: **set KSR of operator.**
- x is the operator, either 1 or 2. a value of 0 means "all operators".
- y determines whether KSR is on.

## info

---

this chip uses the [FM \(OPLL\)](#)<sub>142</sub> instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Patch set:** changes the chip model, providing different built-in sounds.

the following options are visible in drums mode:

- **Ignore top/hi-hat frequency changes:** in drums mode, makes the top/hi-hat channels not write frequency since they share it with snare and tom.
- **Apply fixed frequency to all drums at once:** sets the frequency of all drums to that of a fixed frequency OPLL drums instrument when one note with it is reached.

# Yamaha OPZ (YM2414)

---

**disclaimer: despite the name, this has nothing to do with teenage engineering's OP-Z synth!**

this is the YM2151's little-known successor, used in the Yamaha TX81Z and a few other Yamaha synthesizers. oh, and the Korg Z3 too.

it adds these features on top of the YM2151:

- 8 waveforms (but they're different from the OPL ones)
- per-channel (possibly) linear volume control separate from TL
- increased multiplier precision (in 1/16ths)
- 4-step envelope generator shift (minimum TL)
- another LFO
- no per-operator key on/off
- fixed frequency mode per operator (kind of like OPN family's extended channel mode but with a bit less precision and for all 8 channels)
- "reverb" effect (actually extends release)

unlike the YM2151, this chip is officially undocumented. very few efforts have been made to study the chip and document it...

therefore emulation of this chip in Furnace is incomplete and uncertain.

no plans have been made for TX81Z MIDI passthrough, because:

- Furnace works with register writes rather than MIDI commands
- the MIDI protocol is slow (would not be enough).
- the TX81Z is very slow to process a note on/off or parameter change event.
- the TL range has been reduced to 0-99, but the chip goes from 0-127.

## effects

---

- 10xx: **set noise frequency of channel 8 operator 4.** 00 disables noise while 01 to 20 enable it.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value..
- 17xx: **set LFO speed.**
- 18xx: **set LFO waveform.** xx may be one of the following:
- 00: saw
- 01: square
- 02: triangle
- 03: noise
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**

- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 1Exx: **set LFO AM depth.**
- 1Fxx: **set LFO PM depth.**
- 24xx: **set LFO 2 speed.**
- 25xx: **set LFO 2 waveform.** xx may be one of the following:
  - 00: saw
  - 01: square
  - 02: triangle
  - 03: noise
- 26xx: **set LFO 2 AM depth.**
- 27xx: **set LFO 2 PM depth.**
- 28xy: **set reverb of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value.
- 2Axy: **set waveform of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value.
- 2Bxy: **set EG shift of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value.
- 2Cxy: **set fine multiplier of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value.
- 2Fxx: **enable envelope hard reset.**
  - this works by inserting a quick release and tiny delay before a new note.
- 3xyy: **set fixed frequency of operator 1/2.**
  - x is the block (0-7 for operator 1; 8-F for operator 2).
  - y is the frequency. fixed frequency mode will be disabled if this is less than 8.
  - the actual frequency is:  $y * (2^x)$ .
- 4xyy: **set fixed frequency of operator 3/4.**
  - x is the block (0-7 for operator 3; 8-F for operator 4).
  - y is the frequency. fixed frequency mode will be disabled if this is less than 8.
  - the actual frequency is:  $y * (2^x)$ .
- 50xy: **set AM of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y determines whether AM is on.
- 51xy: **set SL of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value.
- 52xy: **set RR of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value.
- 53xy: **set DT of operator.**
  - x is the operator (1-4). a value of 0 means "all operators".
  - y is the value:
    - 0: +0
    - 1: +1
    - 2: +2
    - 3: +3

- 4: -0
- 5: -3
- 6: -2
- 7: -1
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set DT2 of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxx: **set D2R/SR of all operators.**
- 5Cxx: **set D2R/SR of operator 1.**
- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 61xx: **set algorithm** (0 to 7).
- 62xx: **set LF0 FM depth** (0 to 7).
- 63xx: **set LF0 AM depth** (0 to 3).
- 64xx: **set LF02 FM depth** (0 to 7).
- 65xx: **set LF02 AM depth** (0 to 3).

## info

---

this chip uses the [FM \(OPZ\)](#)<sup>151</sup> instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Pseudo-PAL:** run the chip on a PAL clock. such a configuration has never been used in hardware.
- **Broken pitch macro/slides:** due to an oversight, pitch slides were twice as fast in older versions of Furnace. this option exists for compatibility.

# PC Engine/TurboGrafx-16

---

a console from NEC that, depending on a region:

- attempted to enter the fierce battle between Nintendo and Sega, but because its capabilities are a mix of third and fourth generation, it failed to last long (US and Europe), or
- was Nintendo's most fearsome rival, completely defeating Sega Mega Drive and defending it self against Super Famicom (Japan).

it has 6 wavetable channels and the last two ones also double as noise channels.  
furthermore, it has some PCM and LFO!

## effects

---

- 10xx: **change wave.**
- 11xx: **toggle noise mode.** only available in the last two channels.
- 12xx: **setup LFO.** the following values are accepted:
  - 00: LFO disabled.
  - 01: LFO enabled, shift 0.
  - 02: LFO enabled, shift 4.
  - 03: LFO enabled, shift 8.
- when LFO is enabled, channel 2 is muted and its output is passed to channel 1's frequency.
- 13xx: **set LFO speed.**
- 17xx: **toggle LEGACY sample mode.**
- **this effect exists only for compatibility reasons! its use is NOT recommended. use Sample type instruments instead.**

## info

---

this chip uses the [PC Engine<sup>172</sup>](#) instrument editor.

## channel status

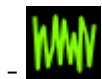
---

the following icons are displayed when channel status is enabled in the pattern view:

- noise mode (channels 5 and 6 only):



off



on



# chip config

---

the following options are available in the Chip Manager window:

- **Pseudo-PAL:** run the chip on a PAL clock. such a configuration has (probably) never existed, despite a planned official PAL version of the PC Engine.
- **Disable anti-click:** waveform switching requires a phase reset, which may cause clicks. Furnace uses a wave-position predicting algorithm to minimize these clicks. enable this option to disable it.
- **Chip revision:** sets the chip revision. HuC6280A has less pops.

# PC Speaker

---

40 years of one square beep - and still going! single channel, no volume control...

## real output

---

so far this is the only chip in Furnace which has a real hardware output option.  
to enable it, select file > configure chip... > PC Speaker > Use system beeper.

be noted that this will only work on Linux as Windows does not provide any user-space APIs to address the PC speaker directly!

you may configure the output method by going in Settings > Emulation > PC Speaker strategy:

- evdev SND\_TONE: uses input events to control the beeper.
- requires write permission to /dev/input/by-path/platform-pcspkr-event-spkr.
- is not 100% frequency-accurate as SND\_TONE demands frequencies, but Furnace uses raw timer periods...
- KIOCSOUND on /dev/tty1: sends the KIOCSOUND ioctl to control the beeper.
- may require running Furnace as root.
- /dev/port: writes to /dev/port to control the beeper.
- requires read/write permission to /dev/port.
- KIOCSOUND on standard output: sends the KIOCSOUND ioctl to control the beeper.
- requires running Furnace on a TTY.
- outb(): uses the low-level kernel port API to control the beeper.
- requires running Furnace as root, or granting it CAP\_SYS\_RAWIO to the Furnace executable:  
`sudo setcap cap_sys_rawio=ep ./furnace.`

real hardware output only works on BIOS/UEFI (non-Mac) x86-based machines! attempting to do this under any other device **will not work**, or may even brick the device (if using /dev/port or outb())!

oh, and of course you also need the beeper to be present in your machine. some laptops connect the beeper output to the built-in speakers (or the audio output jack), and some other don't do this at all.

## effects

---

ha! effects...

## info

---

this chip uses the [Beeper](#)<sub>126</sub> instrument editor.

# chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Speaker type:** select which speaker to use:
- **Unfiltered:** raw square wave.
- **Cone:** filter it to simulate the sound of a cone speaker.
- **Piezo:** simulate the tiny speaker present in most PCs from the 2000s.
- **Use system beeper:** use the actual PC speaker in your machine for output. only works on Linux!
- **Reset phase on frequency change:** reset phase every time the frequency changes. many modern motherboards tend to do this.

# Commodore PET

---

a computer from 1977 which was leader on US schools back then. subsequently the Apple II took its throne.

maybe no better than a computer terminal, but somebody discovered a way to update the screen at turbo rate - and eventually its sound "chip" (it was nothing more than an 8-bit shift register) was abused as well.

some of these didn't even have sound...

## effects

---

- 10xx: **set waveform.**
- xx is a bitmask.

## info

---

this chip uses the [PET<sub>173</sub>](#) instrument editor.

# Pokémon Mini

---

the Pokémon Mini is a ridiculously small handheld system from 2001. its single pulse channel has only three volume steps (full, half, and off)... but variable pulse width.

## effects

---

none.

## info

---

this chip uses the [Pokémon Mini/QuadTone<sup>174</sup>](#) instrument editor.

# POKEY

---

a sound and input chip developed by Atari for their 8-bit computers (Atari 400, 800, XL/XE and so on). 4 channels of signature Atari sounds.

## effects

---

- 10xx: **set waveform.**
- 0: harsh noise (poly5+17)
- 1: square buzz (poly5)
- 2: weird noise (poly4+5)
- 3: square buzz (poly5)
- 4: soft noise (poly17)
- 5: square
- 6: bass (poly4)
- 7: buzz (poly4)
- 11xx: **set AUDCTL.** xx is a bitmask.
- bit 7: 9-bit poly mode. shortens noise.
- bit 6: high channel 1 clock (~1.79MHz on NTSC).
  - overrides 15KHz mode.
- bit 5: high channel 3 clock (~1.79MHz on NTSC).
  - overrides 15KHz mode.
- bit 4: join channels 1 and 2 for a wide period range.
  - use with conjunction with bit 6.
  - channel 2 becomes inaccessible when this is on.
- bit 3: join channels 3 and 4 for a wide period range.
  - use with conjunction with bit 5.
  - channel 4 becomes inaccessible when this is on.
- bit 2: high-pass filter (channels 1 and 3).
  - filtered output on channel 1 (I suggest you to set channel 3 volume to 0).
  - use for PWM effects (not automatic!).
- bit 1: high-pass filter (channels 2 and 4).
  - filtered output on channel 2 (I suggest you to set channel 4 volume to 0).
  - use for PWM effects (not automatic!).
- bit 0: 15KHz mode.
- 12xx: **toggle two-tone mode.**
- when enabled, channel 2 modulates channel 1. I don't know how, but it does.
- only on ASAP core.

## info

---

this chip uses the [POKEY<sub>175</sub>](#) instrument editor.

when two channels are joined for filtered output, the channel bar will show `filter` on a bracket tying them together.

when two channels are joined for wide period range, the channel bar will show `16-bit` on a bracket tying them together.

## ROM export

---

a song can be exported as SAP-R, a compressed register dump. more details on the format are available (in Polish) at [its page on the Atariki wiki](http://atariki.krap.pl/index.php/SAP_%28format_pliku%29) ([http://atariki.krap.pl/index.php/SAP\\_%28format\\_pliku%29](http://atariki.krap.pl/index.php/SAP_%28format_pliku%29)) .

## chip config

---

the following options are available in the Chip Manager window:

- **PAL**: run the chip at PAL clock (1.77MHz) instead of NTSC (1.79MHz).

# PowerNoise

---

a fantasy sound chip created by The Beesh-Spweesh! and jvsTSX for the Hexheld fantasy video game console.

its design employs linear-feedback shift registers (LFSR) for sound generation. this technology is used in many random number generators to produce noise, but it is also capable of producing other tones.

it has three noise channels and one "slope" channel capable of generating a variety of saw waves.

it outputs stereo sound with 4-bit volume control per channel and 2-bit master volume control.

refer to [its instrument type's documentation](#)<sup>176</sup> for details on sound synthesis.

## effect commands

---

- 20xx: **load LFSR value (low byte).**
- on the slope channel, this sets its accumulator (from 00 to 7F).
- 21xx: **load LFSR value (high byte).**
- 22xx: **write to I/O port A.**
- 23xx: **write to I/O port B.**

## info

---

this chip uses the [PowerNoise](#)<sup>176</sup> instrument editor.



# Casio PV-1000

---

released only in Japan, this console was pulled after only a few weeks on the market. it has only 3 square waves with 6-bit pitch resolution and no bass.

## effects

---

- 10xx: **set ring modulation.**
- amplitude modulation by the previous channel's output.
- 0 turns it off and 1 turns it on.

## info

---

this chip uses the [PV-1000<sub>179</sub>](#) instrument editor.

# Capcom QSound (DL-1425)

---

this chip was used in Capcom's CP System Dash, CP System II and ZN arcade PCBs.

it supports 16 PCM channels and uses the patented (now expired) QSound stereo expansion algorithm, as the name implies.

because the chip lacks sample interpolation, it is recommended that you try to play samples at around 24038Hz to avoid aliasing. this is especially important for e.g. cymbals.

the QSound chip also has a small echo buffer, somewhat similar to the SNES, although with a very basic (and non-adjustable) filter. it is however possible to adjust the feedback and length of the echo buffer (the initial values can be set in the "configure chip" option in the file menu or the chip manager).

there are also 3 ADPCM channels. ADPCM samples are fixed to 8012Hz.

## effects

---

- 10xx: **set echo feedback level.**
- this effect will apply to all channels.
- 11xx: **set echo level.**
- 12xx: **toggle QSound algorithm.**
- on by default.
- 3xxx: **set echo delay buffer length.**

## info

---

this chip uses the [QSound<sub>180</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Echo delay:** sets the length of the echo buffer.
- **Echo feedback:** sets the amount of feedback on the echo.

# Ricoh RF5C68

---

YM2612's sidekick - poor man's SNES DSP. 8-channel PCM sample-based synthesizer used in Sega CD, Fujitsu FM Towns and some of Sega's arcade machines. supports up to 64KB of sample data.

## effects

---

none so far.

## info

---

this chip uses the [RF5C68<sub>181</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Chip type:** changes the chip model.

# Philips SAA1099

---

this was used by the Game Blaster and SAM Coupé. it's pretty similar to the AY-3-8910, but has stereo sound, twice the channels and two envelopes, both of which are highly flexible. the envelopes work like this:

- an instrument with envelope settings is placed on channel 2 or channel 5
- an instrument that is used as an "envelope output" is placed on channel 3 or channel 6 (you may want to disable wave output on the output channel)

## effects

---

- 10xy: **set channel mode**.
- x toggles noise.
- y toggles square.
- this effect affects either the first 3 or last 3 channels, depending on where it is placed.
- 11xx: **set noise frequency**.
- this effect affects either the first 3 or last 3 channels, depending on where it is placed.
- 12xx: **setup envelope**. this is a bitmask.
- bit 7 toggles the envelope.
- bit 5 toggles whether to use a fixed frequency or lock to the frequency of channel 2 or 5.
- bit 4 sets the envelope resolution.
- bits 1 to 3 set the envelope shape:
  - 000: always off
  - 001: always on
  - 010: down
  - 011: down loop (saw)
  - 100: up down
  - 101: up down loop (triangle)
  - 110: up then off
  - 111: up loop (reverse saw)
- bit 0 sets whether the right output will mirror the left one.
- this effect affects either the first 3 or last 3 channels, depending on where it is placed.

## info

---

this chip uses the [SAA1099<sub>182</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- sets the rate at which the chip will run.

# Konami SCC/SCC+

---

the Sound Creative Chip (SCC) adds 5 channels of wavetable to your MSX!  
it was used in (of course) several Konami games, which had better audio quality due to the extra channels provided by this chip (poor AY since nobody used the envelope for bass).

the only problem? the waveform of the fourth channel is shared with the fifth one due to not enough memory in the chip!  
the SCC+ fixes this issue though (while being compatible with SCC games).

## effects

---

- 10xx: **change wave.**

## info

---

this chip uses the [Konami SCC/Bubble System WSG<sub>184</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# SegaPCM

---

16 channels of PCM? no way!

yep, that's right! 16 channels of PCM!

a chip used in the Sega OutRun/X/Y arcade boards. eventually the MultiPCM surpassed it with 28 channels, and later they joined the software mixing gang.

## 5-channel SegaPCM

---

Furnace also has a five channel version of this chip, but it only exists for DefleMask compatibility reasons (which doesn't expose the other channels for rather arbitrary reasons).

## effects

---

- 20xx: **set PCM frequency.**
- xx is a 256th fraction of 31250Hz.
- this effect exists mostly for DefleMask compatibility; it is otherwise recommended to use Sample type instruments.

## info

---

this chip uses the [SegaPCM<sub>185</sub>](#) instrument editor.

# SID2

---

a fictional chip created by LTVA. the idea is to fix [SID<sub>236</sub>](#) flaws and add more features, but not too much.

unlike SID, it has per-channel volume control, better ADSR envelope which doesn't have bugs, more waveform mixing modes and the ability to play tonal noise waves.

each channel now has its own independent filter. filter cutoff and resonance ranges were extended, as well as the frequency; finally, the chip can hit a B-7 note with default clock speed!

## effects

---

- 10xx: **change wave**. lower 4 bits specify the wave:
  - bit 0: triangle
  - bit 1: saw
  - bit 2: pulse
  - bit 3: noise
- 11xx: **set resonance**. xx may be a value between 00 and FF.
- 12xx: **set filter mode**. the following values are accepted:
  - 00: filter off
  - 01: low pass
  - 02: band pass
  - 03: low+band pass
  - 04: high pass
  - 05: band reject/stop/notch
  - 06: high+band pass
  - 07: all pass
- 13xx: **disable envelope reset for this channel**.
- 14xy: **reset cutoff**:
  - if x is not 0: on new note
  - if y is not 0: now
  - this effect is not necessary if the instrument's cutoff macro is absolute.
- 15xy: **reset duty cycle**:
  - if x is not 0: on new note
  - if y is not 0: now
  - this effect is not necessary if the instrument's duty macro is absolute.
- 16xy: **change additional parameters**.
  - x may be one of the following:
    - 0: attack (y from 0 to F)
    - 1: decay (y from 0 to F)
    - 2: sustain (y from 0 to F)
    - 3: release (y from 0 to F)
    - 4: ring modulation (y is 0 or 1)
    - 5: oscillator sync (y is 0 or 1)
    - 6: filter mode (y is 0 to 7)
    - 7: waveform mix mode (y is 0 to 3)
    - 8: noise mode (y is 0 to 3)

- 9: phase reset (y is a discarded parameter and does not matter)
- A: envelope key on/key off (y is 0 (trigger envelope release) or 1 (restart envelope again))
- B: filter on/off (y is 0 (disable filter) or 1 (enable filter))
- 17xx: **pulse width slide up.**
- xx is speed. if it is 00, the slide is stopped.
- 18xx: **pulse width slide down.**
- xx is speed. if it is 00, the slide is stopped.
- 19xx: **filter cutoff slide up.**
- xx is speed. if it is 00, the slide is stopped.
- 1Axx: **filter cutoff slide down.**
- xx is speed. if it is 00, the slide is stopped.
- 3xxx: **set duty cycle.** xxx range is 000 to FFF.
- 4xxx: **set cutoff.** xxx range is 000 to FFF.

## info

---

this chip uses the [SID2<sub>186</sub>](#) instrument editor.



# SID3

---

a fictional chip created by LTVA. the idea is to stay vaguely in the [SID<sub>236</sub>](#)-like category of chips, but add a lot of features and more channels.

the chip has 6 synth channels and one channel capable of playing wavetable or streamed samples.

each of synth channels has the following:

- two phase accumulator based oscillators, one for tone and one for noise LFSR clocking; frequency range is from 0.001Hz to around 15kHz at default clock speed.
- 5 waveform types which can be enabled in any combination: pulse (with 16-bit pulse width control), triangle, sawtooth, noise and so called special wave.
- there are 58 different special waves, including all [OPL<sub>3</sub><sub>268</sub>](#) and [OPZ<sub>273</sub>](#) waveforms, their clipped versions, cubed sawtooth and triangle variations, and more...
- noise is generated from 30-bit LFSR. like in C64, eight of the bits are used to form 8-bit noise signal. user can adjust feedback freely, any bit can be a feedback bit. some feedback bits' configurations produce very short looped noise which is perceived as tone. see the [SID3 instrument description<sub>188</sub>](#) for notable feedback bit configurations which are automatically detected by Furnace: upon detection noise frequency is adjusted in such a way that fundamental frequency of such tonal noise becomes the note frequency the channel is currently playing (noise stays in tune).
- 1-bit noise mode is available for [AY<sub>228</sub>](#) fans. in this mode the highest LFSR bit is read as output. by rapidly switching between usual and 1-bit noise modes one can produce interesting rattling-like percussive sound.
- 5 waveform mixing modes: 8580 SID (C64's combined waves; mode does bitwise AND with noise and special wave), bitwise AND, bitwise OR, bitwise XOR and sum of oscillators' signals.
- hard sync between channels. each channel can have any other channel as sync source, even itself.
- ring modulation between channels. each channel can have any other channel as modulation source, even itself. when you self-modulate, you effectively square the signal, but the behavior is a bit different.
- phase modulation between channels. each channel can have any other channel as modulation source, even itself. when you self-modulate, you have an effect similar to enabling strong feedback. channel output after filters is used as modulation source.
- ADSR envelope with sustain rate setting (how fast sound decays when envelope is in sustain phase).
- 4 independent filters. each filter has its own cutoff, resonance, output volume, mode, on/off, and distortion setting. each filter can be connected to channel's ADSR output. each filter's output can be routed to the final channel output. each filter output can be connected to each filter's input (full connection matrix).
- distortion is a simple asymmetrical distortion with hyperbolic tangent function for positive half of the wave and exponential function for negative half.
- several filters can be chained for flexible subtractive synth or to increase filter's slope (which is 12 dB/octave for a single filter).
- multiple filter modes can be selected simultaneously. for example, selecting both "low" and "high" results in a bandstop (notch) filter.
- adjustable feedback. feedback saves two previous channel's outputs and adds them to an accumulator on the next step before computing the waveform signal.

- fine control over left and right channel panning.
- left and right channels' signals can be inverted to create simple "surround" sound.

ADSR can be reset to the start of attack phase. phase of tone and noise oscillators can also be reset, and with noise oscillator reset noise LFSR is also reset to initial state.

wave channel has all these features, except, obviously, waveform generation stage, as well as feedback and noise generator.

## effects

---

- 1xxx: **set filter 1 cutoff.** xxx range is 000 to FFF.
- 2xxx: **set filter 2 cutoff.** xxx range is 000 to FFF.
- 3xxx: **set filter 3 cutoff.** xxx range is 000 to FFF.
- 4xxx: **set filter 4 cutoff.** xxx range is 000 to FFF.
- 5xxx: **set duty cycle.** xxx range is 000 to FFF.
- 60xx: **change wave.** lower 5 bits specify the wave:
  - bit 0: triangle
  - bit 1: saw
  - bit 2: pulse
  - bit 3: noise
  - bit 4: special wave
- 61xx: **change special wave.** xx range is 00 to 39.
- 62xx: **modulation control.** lower 3 bits control the modulation:
  - bit 0: ring modulation
  - bit 1: oscillator sync
  - bit 2: phase modulation
- 63xy: **reset duty cycle:**
  - if x is not 0: on new note
  - if y is not 0: now
- 64xx: **set ring modulation source channel.** xx range is 00 to 07 where 07 means self-modulation and lower values specify a source channel.
- 65xx: **set hard sync source channel.** xx is 00 to 06.
- 66xx: **set phase modulation source channel.** xx is 00 to 06.
- 67xx: **set attack.** xx range is 00 to FF.
- 68xx: **set decay.** xx range is 00 to FF.
- 69xx: **set sustain level.** xx range is 00 to FF.
- 6Axx: **set sustain rate.** xx range is 00 to FF.
- 6Bxx: **set release.** xx range is 00 to FF.
- 6Cxx: **set waveform mix mode.** xx range is 00 to 04.
- 6Dxx: **set noise LFSR feedback bits (lower byte).** xx range is 00 to FF.
- 6Exx: **set noise LFSR feedback bits (medium byte).** xx range is 00 to FF.
- 6Fxx: **set noise LFSR feedback bits (higher byte).** xx range is 00 to FF.
- 70xx: **set noise LFSR feedback bits (highest bits).** xx range is 00 to 3F.
- 71xx: **set filter 1 resonance.** xx range is 00 to FF.
- 72xx: **set filter 2 resonance.** xx range is 00 to FF.
- 73xx: **set filter 3 resonance.** xx range is 00 to FF.
- 74xx: **set filter 4 resonance.** xx range is 00 to FF.
- 75xx: **set noise/wave channel mode.** xx range is 00 to 01.
  - on synth channels 00 sets usual noise mode and 01 sets 1-bit noise mode.

- on wave channel 00 sets wavetable mode and 01 sets streamed PCM sample playback mode.
- 76xx: **set filter 1 output volume.** xx range is 00 to FF.
- 77xx: **set filter 2 output volume.** xx range is 00 to FF.
- 78xx: **set filter 3 output volume.** xx range is 00 to FF.
- 79xx: **set filter 4 output volume.** xx range is 00 to FF.
- 7Axx: **set filter 1 distortion.** xx range is 00 to FF.
- 7Bxx: **set filter 2 distortion.** xx range is 00 to FF.
- 7Cxx: **set filter 3 distortion.** xx range is 00 to FF.
- 7Dxx: **set filter 4 distortion.** xx range is 00 to FF.
- 7Exx: **set feedback.** xx range is 00 to FF.
- 7Fxx: **channel inversion control.** lower 2 bits control the channel signal inversion:
  - bit 0: invert right channel
  - bit 1: invert left channel
- A0xy: **set filter mode.** x is the filter (0-3), and lower 3 bits of y control the mode:
  - bit 0: low pass
  - bit 1: band pass
  - bit 2: high pass
- A1xy: **set filter connection.** x is the filter (0-3), and lower 2 bits of y control the connection:
  - bit 0: connect filter input to channel's ADSR output
  - bit 1: connect filter's output to final channel output
- A2xy: **set filter connection matrix row.** x is the filter (0-3), and lower 4 bits of y control the inter-filter connections:
  - bit 0: connect filter input to filter 1 output
  - bit 1: connect filter input to filter 2 output
  - bit 2: connect filter input to filter 3 output
  - bit 3: connect filter input to filter 4 output
- A3xy: **enable filter.** x is the filter (0-3), y is either 0 (filter disabled) or 1 (filter enabled).
- A4xx: **pulse width slide up.** xx is speed. A400 stops the slide.
- A5xx: **pulse width slide down.** xx is speed. A500 stops the slide.
- A6xx: **filter 1 cutoff slide up.** xx is speed. A600 stops the slide.
- A7xx: **filter 1 cutoff slide down.** xx is speed. A700 stops the slide.
- A8xx: **filter 2 cutoff slide up.** xx is speed. A800 stops the slide.
- A9xx: **filter 2 cutoff slide down.** xx is speed. A900 stops the slide.
- AAXx: **filter 3 cutoff slide up.** xx is speed. AA00 stops the slide.
- ABxx: **filter 3 cutoff slide down.** xx is speed. AB00 stops the slide.
- ACxx: **filter 4 cutoff slide up.** xx is speed. AC00 stops the slide.
- ADxx: **filter 4 cutoff slide down.** xx is speed. AD00 stops the slide.
- AExx: **tone phase reset.** xx is the tick on which the phase reset happens.
- AFxx: **noise phase reset.** xx is the tick on which the phase reset happens.
- B0xx: **envelope reset.** xx is the tick on which the envelope reset happens.
- B1xy: **filter cutoff scaling control.** x is the filter (0-3), and lower 2 bits of y control the scaling:
  - bit 0: enable cutoff scaling
  - bit 1: inverse cutoff scaling
- B2xy: **filter resonance scaling control.** x is the filter (0-3), and lower 2 bits of y control the scaling:
  - bit 0: enable resonance scaling
  - bit 1: inverse resonance scaling

## info

---

this chip uses the [SID3<sub>188</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Quarter clock speed:** make chip run at quarter the default clock rate (1MHz is default). this lowers CPU load almost 4 times at the cost of filters becoming unstable or having different timbre at high cutoff and resonance settings. this option affects the chip only in playback mode. when rendering a module into an audio file, this option is not applied.

# Sharp SM8521

---

the SM8521 is the CPU and sound chip of the Game.com, a handheld console released in 1997 as a competitor to the infamous Nintendo Virtual Boy.

sadly, the Game.com ended up being a failure as well, mostly due to poor quality games. the Game.com only lasted 3 years before being discontinued.

however, for its time, it was a pretty competitively priced system. the Game Boy Color was to be released in a year for \$79.95, while the Game.com was released for \$69.99; its later model, the Pocket Pro, was released in mid-1999 for \$29.99 due to the Game.com's apparent significant decrease in value.

in fact, most games never used the wavetable/noise mode of the chip. Sonic Jam, for example, uses a sine wave with a software-controlled volume envelope on the DAC channel (see below for more information on the DAC channel).

the sound-related features and quirks of the SM8521 are as follows:

- 2 4-bit wavetable channels
- a noise channel (which can go up to a very high pitch, creating an almost periodic noise sound)
- 5-bit volume
- a low bit-depth output (which means it distorts a lot).
- it phase resets when you switch waves
- 12-bit pitch with a wide frequency range
- a software-controlled D/A register that (potentially) requires all other registers to be stopped to play. due to this, it is currently not implemented in Furnace.

## effects

---

- 10xx: **set waveform.**
- xx is a value between 0 and 255 that sets the waveform of the channel you place it on.

## info

---

this chip uses the [SM8521<sup>192</sup>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# TI SN76489 (e.g. Sega Master System)

---

a relatively simple sound chip made by Texas Instruments. a derivative of it is used in Sega's Master System, the predecessor to Genesis. it has three square wave channels and one noise channel... not really.

nominal mode of SN76489 has 3 square wave channels, with noise channel having only 3 preset frequencies to use (absurdly low, very low, low). to use more pitches, one can enable a mode which "steals" the frequency from square wave channel 3. by doing that, SN76489 becomes effectively a 3 channel sound chip. in addition, periodic noise mode can be enabled, with same caveats.

the original iteration of the SN76489 used in the TI-99/4A computer, the SN94624, could only produce tones as low as 100Hz, and was clocked at 447 KHz. all later versions (such as the one in the Master System and Genesis) had a clock divider but ran on a faster clock, making a chip **very** high pitched... except for the SN76494, which can play notes as low as 13.670 Hz (A -1). as a result, its pitch accuracy for higher notes is compromised.

## SN7 versions

---

SN7 was extremely popular due to low cost. therefore, it was cloned and copied to no end, often with minor differences between each other. Furnace supports several of these:

- SN94624, can only produce tones as low as 100Hz, and is clocked at 447 KHz.
- SN76494, which can play notes as low as 13.670 Hz (A -1). it has a different noise feedback and invert masks.
- SN76489, identical to SN94624, just without a clock divider.
- SN76489A, identical to 76494, just with a /8 clock divider.
- SN76496, literally identical to former. why is it even here?
- SN76496 with a Atari-like short noise. the chip of many legend and rumours which may be a result of inaccurate emulation.
- Sega Master System VDP version has a different, characteristic noise LFSR.
- Game Gear SN7, identical to the above, but with stereo.
- NCR8496, different noise invert mask.
- PSSJ3, literally identical to the former. it just swaps "high" and "low" signals in the output, which results in no audible difference.

## effects

---

- 20xy: **set noise mode.**
- x controls whether to inherit frequency from channel 3.
  - 0: use one of 3 preset frequencies (C: A-2; C#: A-3; D: A-4).
  - 1: use frequency of channel 3.
- y controls whether to select noise or thin pulse.
  - 0: thin pulse.
  - 1: noise.

## info

---

this chip uses the [SN76489/Sega PSG<sub>178</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Chip type:** changes the chip type. see above for more details.
- **Disable noise period change phase reset:** when enabled, the noise channel won't be reset every time its frequency changes. very useful.
- **Disable easy period to note mapping on upper octaves:** Furnace maps the notes in the upper octaves to periods, for easier noise tuning. this option allows you to disable this feature.

# Super Nintendo Entertainment System (SNES) / Super Famicom

---

the successor to NES to compete with Genesis, packing superior graphics and sample-based audio.

its Sony-developed audio system features a DSP chip, SPC700 CPU, and 64KB of dedicated SRAM used by both.

this whole system itself is pretty much a separate computer that the main CPU needs to upload its program and samples to.

Furnace communicates with the DSP directly and provides a full 64KB of memory. this memory might be reduced excessively on ROM export to make up for playback engine and pattern data. you can go to window > statistics to see how much memory your samples are using.

some notable features of the DSP are:

- pitch modulation.
- a built in noise generator, useful for hi-hats, cymbals, rides, effects, among other things.
- per-channel echo, which unfortunately eats up a lot of memory but can be used to save channels in songs.
- an 8-tap FIR filter for the echo, which is basically a procedural low-pass filter that you can edit however you want.
- sample loop, but the loop points have to be multiples of 16.
- left/right channel invert for surround sound.
- ADSR and gain envelope modes.
- 7-bit volume per channel.
- sample interpolation, which is basically a low-pass filter that gets affected by the pitch of the channel.

Furnace also allows the SNES to use wavetables (and the wavetable synthesizer) in order to create more 'animated' sounds, using less memory than regular samples.

## effects

---

- 10xx: **set waveform.**
- 11xx: **toggle noise mode.**
- 12xx: **toggle echo on this channel.**
- 13xx: **toggle pitch modulation.** frequency modulation by the previous channel's output. no effect on channel 1.
- 14xy: **toggle inverting the left or right channels.** x is left, y is right.
- 15xx: **set envelope mode.** see gain chart below for 1 through 5.
- 0: ADSR mode.
- 1: gain (direct). volume holds at one level.



- 2: linear decrement. volume lowers by subtractions of 1/64.
- 3: exponential decrement. volume lowers by multiplications of 255/256.
- 4: linear increment. volume rises by additions of 1/64.
- 5: bent line (inverse log) increment. volume rises by additions of 1/64 until 3/4, then additions of 1/256.
- 16xx: **set gain**. 00 to 7F if direct, 00 to 1F otherwise.
- 18xx: **enable echo buffer**.
- 19xx: **set echo delay**. range is 0 to F.
- 1Axx: **set left echo channel volume**.
- 1Bxx: **set right echo channel volume**.
- 1Cxx: **set echo feedback**.
- all of these are signed numbers.
- 00 to 7F for 0 to 127.
- 80 to FF for -128 to -1.
- setting these to -128 is not recommended as it may cause echo output to overflow and therefore click.
- 1Dxx: **set noise generator frequency**. range is 00 to 1F. see noise frequencies chart below.
- 1Exx: **set left dry / global volume**.
- 1Fxx: **set right dry / global volume**.
- these do not affect echo.
- 20xx: **set attack**. range is 0 to F.
- 21xx: **set decay**. range is 0 to 7.
- 22xx: **set sustain**. range is 0 to 7.
- 23xx: **set release**. range is 00 to 1F.
- these four are only used in ADSR envelope mode. see ADSR chart below.
- 3xyy: **set echo filter coefficient**.
- x is the coefficient from 0 to 7.
- yy is the value (signed number).
  - 00 to 7F for 0 to 127.
  - 80 to FF for -128 to -1.
- note: be sure the sum of all coefficients is between -128 and 127. sums outside that may result in overflow and therefore clicking.
- see SnesLab for [echo filter explanations and examples](https://sneslab.net/wiki/FIR_Filter#Uses) (https://sneslab.net/wiki/FIR\_Filter#Uses) .

## info

---

this chip uses the [SNES<sub>193</sub>](#) instrument editor.










when two channels are joined for pitch modulation, the channel bar will show mod on a bracket tying them together.

when using sample offset commands, be sure to open each involved sample in the sample editor, look to the "Info" section at the top-left, and check the "no BRR filters" box. this prevents sound glitches, at the cost of lowering the sample quality to 4-bit.

# channel status

---

the following icons are displayed when channel status is enabled in the pattern view:

- envelope/gain status:
  -  direct gain
  -  linear gain decrease
  -  logarithmic gain decrease
  -  linear gain increase
  -  bent-line gain increase
  -  envelope attack
  -  envelope decay
  -  envelope sustain
  -  envelope release

## chip config

---

the following options are available in the Chip Manager window:

- **Volume scale:** scale volumes to prevent clipping/distortion.
- **Enable echo:** enables the echo buffer.
- **Initial echo state:** selects which channels will have echo applied.
- **Delay:** sets echo length.
- **Feedback:** sets how much of the echo output will be fed back into the buffer.
- **Echo volume:** sets echo volume.
- **Echo filter:** adjusts echo filter.
- **Dec/Hex:** toggles decimal or hexadecimal mode for the filter settings text entry box to the right.
- SnesLab provides [echo filter explanations and examples](https://sneslab.net/wiki/FIR_Filter#Uses) ([https://sneslab.net/wiki/FIR\\_Filter#Uses](https://sneslab.net/wiki/FIR_Filter#Uses)) . their example filter strings can be pasted directly into the filter settings text entry box if set to Hex mode.
- **Disable Gaussian interpolation:** removes sample interpolation, resulting in crisper but aliased sound. not accurate to hardware.
- **Anti-click:** reduces clicks in the output by using hardware envelopes to smooth them out.
- make sure your samples start on center, or else you will still hear clicks.

# ADSR

ATTACK	0→1 TIME	DECAY	1→S TIME	SUSTAIN	RATIO	RELEASE	S→0 TIME
00	4.1s	00	1.2s	00	1/8	00	∞
01	2.5s	01	740ms	01	2/8	01	38s
02	1.5s	02	440ms	02	3/8	02	28s
03	1.0s	03	290ms	03	4/8	03	24s
04	640ms	04	180ms	04	5/8	04	19s
05	380ms	05	110ms	05	6/8	05	14s
06	260ms	06	74ms	06	7/8	06	12s
07	160ms	07	37ms	07	1	07	9.4s
08	96ms					08	7.1s
09	64ms					09	5.9s
0A	40ms					0A	4.7s
0B	24ms					0B	3.5s
0C	16ms					0C	2.9s
0D	10ms					0D	2.4s
0E	6ms					0E	1.8s
0F	0ms					0F	1.5s
						10	1.2s
						11	880ms
						12	740ms
						13	590ms
						14	440ms
						15	370ms
						16	290ms
						17	220ms
						18	180ms
						19	150ms
						1A	110ms
						1B	92ms
						1C	74ms
						1D	55ms
						1E	37ms
						1F	18ms

reference: [Super Famicom Development Wiki](https://wiki.superfamicom.org/spc700-reference#dsp-voice-register:-adsr-1097) (https://wiki.superfamicom.org/spc700-reference#dsp-voice-register:-adsr-1097)

# gain

VALUE	LINEAR INC.	BENT LINE INC.	LINEAR DEC.	EXPONENT DEC.
00	∞	∞	∞	∞
01	4.1s	7.2s	4.1s	38s
02	3.1s	5.4s	3.1s	28s
03	2.6s	4.6s	2.6s	24s
04	2.0s	3.5s	2.0s	19s
05	1.5s	2.6s	1.5s	14s
06	1.3s	2.3s	1.3s	12s
07	1.0s	1.8s	1.0s	9.4s
08	770ms	1.3s	770ms	7.1s
09	640ms	1.1s	640ms	5.9s
0A	510ms	900ms	510ms	4.7s
0B	380ms	670ms	380ms	3.5s
0C	320ms	560ms	320ms	2.9s
0D	260ms	450ms	260ms	2.4s
0E	190ms	340ms	190ms	1.8s
0F	160ms	280ms	160ms	1.5s
10	130ms	220ms	130ms	1.2s
11	96ms	170ms	96ms	880ms
12	80ms	140ms	80ms	740ms
13	64ms	110ms	64ms	590ms
14	48ms	84ms	48ms	440ms
15	40ms	70ms	40ms	370ms
16	32ms	56ms	32ms	290ms
17	24ms	42ms	24ms	220ms
18	20ms	35ms	20ms	180ms
19	16ms	28ms	16ms	150ms
1A	12ms	21ms	12ms	110ms
1B	10ms	18ms	10ms	92ms
1C	8ms	14ms	8ms	74ms
1D	6ms	11ms	6ms	55ms
1E	4ms	7ms	4ms	37ms
1F	2ms	3.5ms	2ms	18ms

reference: [Super Famicom Development Wiki](https://wiki.superfamicom.org/spc700-reference#dsp-voice-register:-gain-1156) (https://wiki.superfamicom.org/spc700-reference#dsp-voice-register:-gain-1156)

## noise frequencies

---

VALUE	FREQ.	VALUE	FREQ.
00	0 Hz	10	500 Hz
01	16 Hz	11	667 Hz
02	21 Hz	12	800 Hz
03	25 Hz	13	1.0 KHz
04	31 Hz	14	1.3 KHz
05	42 Hz	15	1.6 KHz
06	50 Hz	16	2.0 KHz
07	63 Hz	17	2.7 KHz
08	83 Hz	18	3.2 KHz
09	100 Hz	19	4.0 KHz
0A	125 Hz	1A	5.3 KHz
0B	167 Hz	1B	6.4 KHz
0C	200 Hz	1C	8.0 KHz
0D	250 Hz	1D	10.7 KHz
0E	333 Hz	1E	16 KHz
0F	400 Hz	1F	32 KHz

reference: [Super Famicom Development Wiki](https://wiki.superfamicom.org/spc700-reference#dsp-register:-flg-1318) (<https://wiki.superfamicom.org/spc700-reference#dsp-register:-flg-1318>)

## resources

---

- [SNES-format BRR samples](https://www.smwcentral.net/?p=stion&s=brrsamples) (<https://www.smwcentral.net/?p=stion&s=brrsamples>) at SMW Central

# tildearrow Sound Unit

---

a fantasy sound chip, used in the specs2 fantasy computer designed by tildearrow.

it has the following capabilities:

- 8 channels of either waveform or sample
- stereo sound
- 8 waveforms (pulse, saw, sine, triangle, noise, periodic noise, XOR sine and XOR triangle)
- 128 widths for the pulse wave
- per-channel resonant filter
- ring modulation
- volume, frequency and cutoff sweep units (per-channel)
- phase reset timer (per-channel)

## effects

---

- 10xx: **set waveform.**
- 0: pulse wave
- 1: sawtooth
- 2: sine wave
- 3: triangle wave
- 4: noise
- 5: periodic noise
- 6: XOR sine
- 7: XOR triangle
- 12xx: **set pulse width.** range is 0 to 7F.
- 13xx: **set resonance of filter.** range is 0 to FF.
- despite what the internal effects list says (0 to F), you can use a resonance value from 0 to FF (255).
- 14xx: **set filter mode and ringmod.**
- bit 0: ring mod
- bit 1: low pass
- bit 2: high pass
- bit 3: band pass
- 15xx: **set frequency sweep period low byte.**
- 16xx: **set frequency sweep period high byte.**
- 17xx: **set volume sweep period low byte.**
- 18xx: **set volume sweep period high byte.**
- 19xx: **set cutoff sweep period low byte.**
- 1Axx: **set cutoff sweep period high byte.**
- 1Bxx: **set frequency sweep boundary.**
- 1Cxx: **set volume sweep boundary.**
- 1Dxx: **set cutoff sweep boundary.**
- 1Exx: **set phase reset period low byte.**
- 1Fxx: **set phase reset period high byte.**
- 20xx: **toggle frequency sweep.**
- bit 0-6: speed

- bit 7: up direction
- 21xx: **toggle volume sweep**.
- bit 0-4: speed
- bit 5: up direction
- bit 6: loop
- bit 7: alternate
- 22xx: **toggle cutoff sweep**.
- bit 0-6: speed
- bit 7: up direction
- 4xxx: **set cutoff**. range is 0 to FFF.

## info

---

this chip uses the [Sound Unit](#)<sup>195</sup> instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **CPU rate**: sets the rate at which the chip will run.
- **Sample memory**: sets the amount of memory available for samples.
- **DAC resolution**: sets output resolution.
- **Enable echo**: guess.
- **Swap echo channels**: puts left into right and vice-versa.
- **Echo delay**: set echo time.
- **Echo resolution**: set echo resolution. sacrifices quality for time.
- **Echo feedback**: guess.
- **Echo volume**: yep, guess again.

# Toshiba T6W28

---

an enhanced SN76489 derivative. same 4 channels, but with stereo (soft panning!) and noise frequency being fully independent of channel 3's.

this chip was used in Neo Geo Pocket.

## effects

---

- 20xx: **set noise mode.**
- 0: thin pulse.
- 1: noise.

## info

---

this chip uses the [T6W28](#)<sub>197</sub> instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Disable easy period to note mapping on upper octaves:** Furnace maps the notes in the upper octaves to periods, for easier noise tuning. this option allows you to disable this feature.



# MOS Technology TED

---

also called 7360/8360, TED stands for Text Editing Device. it's both a video and audio chip of Com modore budget computers, like Plus/4 and 16.

its audio portion is pretty barren - only 2 channels. one can output square wave and other may be either square or noise.

pitch range is limited as well, akin to that of SN76489, and volume control is global.

## effects

---

none so far.

## info

---

this chip uses the [TED<sub>198</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Global parameter priority:** change the priority of macros which control global parameters, such as volume.
- Left to right: process channels from 1 to 3. the last one to run macros will take effect.
- Last used channel: process channels from oldest to newest (since last note). the one which had the latest note on will take a effect.

# Atari 2600 (TIA)

---

help me! I can't even get this character in where I want!

I've spent hours debugging the issue, counting every possible cycle and I still cannot get it right!

only 2 channels and 31 frequencies?!

Furnace isn't complete without this one...

## effects

---

- 10xx: **select shape.**
- 0: nothing
- 1: buzzy
- 2: low buzzy
- 3: flangy
- 4: square
- 5: square
- 6: pure buzzy
- 7: reedy
- 8: noise
- 9: reedy
- A: pure buzzy
- B: nothing
- C: low square
- D: low square
- E: low pure buzzy
- F: low reedy

## ROM export

---

a song can be exported to assembly code for use with the TlunA software driver for the Atari 2600. see [TlunA on GitHub](https://github.com/AYCEdemo/twin-tiuna) (<https://github.com/AYCEdemo/twin-tiuna>) for explanations of the export options.

## chip config

---

the following options are available in the Chip Manager window:

- **Software pitch driver:** use TlunA, a software pitch driver similar to TIATune. it increases pitch precision by rapidly switching between two pitches.
- **Old pitch table:** use an older method to calculate pitch. only for compatibility.
- **Mixing mode:** changes mixing mode.
- **Mono:** normal output.
- **Mono (no distortion):** process each channel separately to eliminate distortion.
- **Stereo:** output two channels on left and right.
- **PAL:** run slower blah blah blah

# info

this chip uses the [TIA<sub>199</sub>](#) instrument editor.

the arp macro's fixed mode operates differently, writing the direct pitch to the chip. here's a list of pitches.

## shape 1

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	0	2096.0	C-7	+2	2080.0	C-7	-1
	1	1048.0	C-6	+2	1040.0	C-6	-1
	2	698.7	F-5	0.0	693.3	F-5	-1
	3	524.0	C-5	+2	520.0	C-5	-1
	4	419.2	G#4	+16	416.0	G#4	+3
	5	349.3	F-4	0.0	346.7	F-4	-13
	6	299.4	D-4	+33	297.1	D-4	+20
	7	262.0	C-4	+3	260.0	C-4	-11
	8	232.9	A#3	-2	231.1	A#3	-15
	9	209.6	G#3	+15	208.0	G#3	+2
	10	190.5	F#3	+50	189.1	F#3	+37
	11	174.7	F-3	+1	173.3	F-3	-13
	12	161.2	E-3	-39	160.0	D#3	+48
	13	149.7	D-3	+33	148.6	D-3	+20
	14	139.7	C#3	+13	138.7	C#3	+1
	15	131.0	C-3	+3	130.0	C-3	-11
	16	123.3	B-2	-3	122.4	B-2	-16
	17	116.4	A#2	0.0	115.6	A#2	-14
	18	110.3	A-2	+5	109.5	A-2	-8
	19	104.8	G#2	+16	104.0	G#2	+3
	20	99.8	G-2	+31	99.0	G-2	+17
	21	95.3	G-2	-49	94.5	F#2	+36
	22	91.1	F#2	-27	90.4	F#2	-40
	23	87.3	F-2	0.0	86.7	F-2	-12
	24	83.8	E-2	+29	83.2	E-2	+16
	25	80.6	E-2	-39	80.0	D#2	+48
	26	77.6	D#2	-5	77.0	D#2	-18
	27	74.9	D-2	+34	74.3	D-2	+20
	28	72.3	D-2	-27	71.7	D-2	-41
	29	69.9	C#2	+15	69.3	C#2	0

PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
30	67.6	C#2	-44	67.1	C-2	+44
31	65.5	C-2	+3	65.0	C-2	-11
#### shapes 2 and 3						
PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
0	67.6	C#2	-44	67.1	C-2	+44
1	33.8	C#1	-42	33.5	C-1	+42
2	22.5	F#0	-46	22.4	F-0	+46
3	16.9	C#0	-44	16.8	C-0	+44
4	13.5			13.4		
5	11.3			11.2		
6	9.7			9.6		
7	8.5			8.4		
8	7.5			7.5		
9	6.8			6.7		
10	6.1			6.1		
11	5.6			5.6		
12	5.2			5.2		
13	4.8			4.8		
14	4.5			4.5		
15	4.2			4.2		
16	4.0			4.0		
17	3.8			3.7		
18	3.6			3.5		
19	3.4			3.4		
20	3.2			3.2		
21	3.1			3.0		
22	3.0			2.9		
23	2.8			2.8		
24	2.7			2.7		
25	2.6			2.6		
26	2.5			2.5		
27	2.4			2.4		
28	2.3			2.3		
29	2.3			2.2		
30	2.2			2.2		
31	2.1			2.1		
#### shapes 4 and 5						

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	0	15720.0	B-9	-9	15600.0	B-9	-23
	1	7860.0	B-8	-9	7800.0	B-8	-23
	2	5240.0	E-8	-11	5200.0	E-8	-25
	3	3930.0	B-7	-10	3900.0	B-7	-23
	4	3144.0	G-7	+4	3120.0	G-7	-9
	5	2620.0	E-7	-11	2600.0	E-7	-25
	6	2245.7	C#7	+21	2228.6	C#7	+8
	7	1965.0	B-6	-9	1950.0	B-6	-23
	8	1746.7	A-6	-13	1733.3	A-6	-27
	9	1572.0	G-6	+4	1560.0	G-6	-9
	10	1429.1	F-6	+39	1418.2	F-6	+25
	11	1310.0	E-6	-11	1300.0	E-6	-25
	12	1209.2	D-6	+49	1200.0	D-6	+36
	13	1122.9	C#6	+22	1114.3	C#6	+8
	14	1048.0	C-6	+2	1040.0	C-6	-11
	15	982.5	B-5	-10	975.0	B-5	-23
	16	924.7	A#5	-15	917.6	A#5	-28
	17	873.3	A-5	-14	866.7	A-5	-27
	18	827.4	G#5	-7	821.1	G#5	-20
	19	786.0	G-5	+4	780.0	G-5	-9
	20	748.6	F#5	+20	742.9	F#5	+7
	21	714.5	F-5	+39	709.1	F-5	+26
	22	683.5	F-5	-38	678.3	E-5	+48
	23	655.0	E-5	-12	650.0	E-5	-25
	24	628.8	D#5	+18	624.0	D#5	+5
	25	604.6	D-5	+49	600.0	D-5	+36
	26	582.2	D-5	-16	577.8	D-5	-29
	27	561.4	C#5	+21	557.1	C#5	+8
	28	542.1	C#5	-40	537.9	C-5	+47
	29	524.0	C-5	+2	520.0	C-5	-11
	30	507.1	B-4	+45	503.2	B-4	+32
	31	491.3	B-4	-9	487.5	B-4	-23
#### shapes 6, 7, 9 and 10							
	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	0	1014.2	B-5	+45	1006.5	B-5	+32
	1	507.1	B-4	+45	503.2	B-4	+32
	2	338.1	E-4	+43	335.5	E-4	+30

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	3	253.5	B-3	+45	251.6	B-3	+32
	4	202.8	G#3	-42	201.3	G-3	+45
	5	169.0	E-3	+43	167.7	E-3	+30
	6	144.9	D-3	-23	143.8	D-3	-37
	7	126.8	B-2	+42	125.8	B-2	+32
	8	112.7	A-2	+42	111.8	A-2	+28
	9	101.4	G#2	-41	100.6	G-2	+45
	10	92.2	F#2	-6	91.5	F#2	-19
	11	84.5	E-2	+43	83.9	E-2	+31
	12	78.0	D#2	+4	77.4	D#2	-9
	13	72.4	D-2	-24	71.9	D-2	-37
	14	67.6	C#2	-44	67.1	C-2	+44
	15	63.4	B-1	+46	62.9	B-1	+32
	16	59.7	A#1	+41	59.2	A#1	+26
	17	56.3	A-1	+39	55.9	A-1	+27
	18	53.4	G#1	+48	53.0	G#1	+35
	19	50.7	G#1	-41	50.3	G-1	+45
	20	48.3	G-1	-25	47.9	G-1	-39
	21	46.1	F#1	-4	45.7	F#1	-20
	22	44.1	F-1	+16	43.8	F-1	+4
	23	42.3	E-1	+44	41.9	E-1	+28
	24	40.6	E-1	-26	40.3	E-1	-39
	25	39.0	D#1	+4	38.7	D#1	-9
	26	37.6	D-1	+41	37.3	D-1	+27
	27	36.2	D-1	-24	35.9	D-1	-38
	28	35.0	C#1	+19	34.7	C#1	+5
	29	33.8	C#1	-42	33.5	C-1	+42
	30	32.7	C-1	0.0	32.5	C-1	-11
	31	31.7	B-0	+44	31.5	B-0	+33
#### shape 8							

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	0	61.5	B-1	-6	61.1	B-1	-18
	1	30.8	B-0	-6	30.5	B-0	-22
	2	20.5	E-0	-8	20.4	E-0	-17
	3	15.4			15.3		
	4	12.3			12.2		
	5	10.3			10.2		

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	6	8.8			8.7		
	7	7.7			7.6		
	8	6.8			6.8		
	9	6.2			6.1		
	10	5.6			5.6		
	11	5.1			5.1		
	12	4.7			4.7		
	13	4.4			4.4		
	14	4.1			4.1		
	15	3.8			3.8		
	16	3.6			3.6		
	17	3.4			3.4		
	18	3.2			3.2		
	19	3.1			3.1		
	20	2.9			2.9		
	21	2.8			2.8		
	22	2.7			2.7		
	23	2.6			2.5		
	24	2.5			2.4		
	25	2.4			2.3		
	26	2.3			2.3		
	27	2.2			2.2		
	28	2.1			2.1		
	29	2.0			2.0		
	30	2.0			2.0		
	31	1.9			1.9		
#### shapes 12 and 13							

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	0	5240.0	E-8	-11	5200.0	E-8	-25
	1	2620.0	E-7	-11	2600.0	E-7	-25
	2	1746.6	A-6	-14	1733.3	A-6	-27
	3	1310.0	E-6	-11	1300.0	E-6	-25
	4	1048.0	C-6	+2	1040.0	C-6	-11
	5	873.3	A-5	-14	866.7	A-5	-27
	6	748.6	F#5	+20	742.9	F#5	+7
	7	655.0	E-5	-12	650.0	E-5	-25
	8	582.2	D-5	-16	577.8	D-5	-29

	PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
	9	524.0	C-5	+2	520.0	C-5	-11
	10	476.4	A#4	+39	472.7	A#4	+23
	11	436.7	A-4	-13	433.3	A-4	-27
	12	403.1	G-4	+48	400.0	G-4	+34
	13	374.3	F#4	+20	371.4	F#4	+6
	14	349.3	F-4	0.0	346.7	F-4	-13
	15	327.5	E-4	-11	325.0	E-4	-25
	16	308.2	D#4	-17	305.9	D#4	-30
	17	291.1	D-4	-16	288.9	D-4	-29
	18	275.8	C#4	-9	273.7	C#4	-22
	19	262.0	C-4	+3	260.0	C-4	-11
	20	249.5	B-3	+18	247.6	B-3	+5
	21	238.2	A#3	+37	236.4	A#3	+24
	22	227.8	A#3	-40	226.1	A-3	+47
	23	218.3	A-3	-14	216.7	A-3	-27
	24	209.6	G#3	+15	208.0	G#3	+2
	25	201.5	G-3	+47	200.0	G-3	+34
	26	194.1	G-3	-17	192.6	G-3	-31
	27	187.1	F#3	+19	185.7	F#3	+6
	28	180.7	F#3	-41	179.3	F-3	+45
	29	174.7	F-3	+1	173.3	F-3	-13
	30	169.0	E-3	+43	167.7	E-3	+30
	31	163.8	E-3	-11	162.5	E-3	-25
#### shapes 14 and 15							
PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT	
0	338.1	E-4	+43	335.5	E-4	+30	
1	169.0	E-3	+43	167.7	E-3	+30	
2	112.7	A-2	+42	111.8	A-2	+28	
3	84.5	E-2	+43	83.9	E-2	+31	
4	67.6	C#2	-44	67.1	C-2	+44	
5	56.3	A-1	+39	55.9	A-1	+27	
6	48.3	G-1	-25	47.9	G-1	-39	
7	42.3	E-1	+44	41.9	E-1	+28	
8	37.6	D-1	+41	37.3	D-1	+27	
9	33.8	C#1	-42	33.5	C-1	+42	
10	30.7	B-0	-11	30.5	B-0	-22	
11	28.2	A-0	+44	28.0	A-0	+31	



PITCH	NTSC	NOTE	CENT	PAL	NOTE	CENT
12	26.0	G#0	0.0	25.8	G#0	-13
13	24.1	G-0	-29	24.0	G-0	-36
14	22.5	F#0	-46	22.4	F-0	+46
15	21.1	E-0	+42	21.0	E-0	+33
16	19.9	D#0	+42	19.7	D#0	+25
17	18.8	D-0	+40	18.6	D-0	+20
18	17.8	C#0	+45	17.7	C#0	+36
19	16.9	C#0	-44	16.8	C-0	+44
20	16.1	C-0	-30	16.0	C-0	-40
21	15.4			15.2		
22	14.7			14.6		
23	14.1			14		
24	13.5			13.4		
25	13.0			12.9		
26	12.5			12.4		
27	12.1			12		
28	11.7			11.6		
29	11.3			11.2		
30	10.9			10.8		
31	10.6			10.5		

reference: [Atari 2600 VCS Sound Frequency and Waveform Guide](http://7800.8bitdev.org/index.php/Atari_2600_VCS_Sound_Frequency_and_Waveform_Guide) ([http://7800.8bitdev.org/index.php/Atari\\_2600\\_VCS\\_Sound\\_Frequency\\_and\\_Waveform\\_Guide](http://7800.8bitdev.org/index.php/Atari_2600_VCS_Sound_Frequency_and_Waveform_Guide))

# VERA

---

this is a video and sound generator chip used in the Commander X16, a modern 8-bit computer created by the 8-Bit Guy.

it has 16 channels of pulse/triangle/saw/noise and one stereo PCM channel.

currently Furnace does not support the PCM channel's stereo mode, though (except for panning).

## effects

---

- 20xx: **set waveform.**
- 0: pulse
- 1: saw
- 2: triangle
- 3: noise
- 22xx: **set duty cycle.**
- range is 0 to 3F.
- EExx: **ZSM synchronization event.**
- xx is the event payload. this has no effect in how the music is played in Furnace, but the ZSMKit library for the Commander X16 interprets these events inside ZSM files and optionally triggers a callback routine. this can be used, for instance, to cause game code to respond to beats or at certain points in the music.

## info

---

this chip uses the [VERA<sub>200</sub>](#) and [Generic Sample<sub>183</sub>](#) instrument editors.

## chip config

---

the following options are available in the Chip Manager window:

- **Chip revision:** sets which revision of the chip to use.
- V 47.0.0 introduces a slightly different volume table.

# Commodore VIC-20

---

the Commodore VIC-20 was Commodore's major attempt at making a personal home computer, and is the precursor to the Commodore 64.

it was also known as the VC-20 in Germany, and the VIC-1001 in Japan.

it has 4 voices that have a limited but wide tuning range, and like the SN76489 and T6W28, the last voice is dedicated to playing noise.

the 3 pulse wave channels also have different octaves that they can play notes on:

- the first channel is the bass channel, and it can play notes from octave 1.
- the next is the 'mid/chord' channel, and it plays notes from octave 2.
- and rather obviously, the 3rd pulse channel is typically the lead channel, can play notes from octave 3.

these channels are not referred as "square" wave channels since a technique to play 15 additional pulse-like waveforms has been discovered long after the VIC-20's release.

## effect commands

---

- 10xx: **switch waveform.** range is 00 to 0F.

## info

---

this chip uses the [VIC<sub>201</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **PAL:** 1.10MHz instead of 1.02MHz.
- **Disable filtering:** bypasses the hardware's lowpass filter.

# Virtual Boy

---

a "portable" video game console made by Nintendo in the '90s.

it supposedly was the beginning of virtual reality... nah, instead it failed to sell well because you use it for 15 minutes and then you get a headache.

its sound generation chip is called Virtual Sound Unit (VSU), a wavetable chip that is a lot like PC Engine, but unlike that, the waves are twice as tall, it doesn't go too low in terms of frequency (~D-2), and the last channel (yep, it has 6 channels) is a noise one.

additionally, channel 5 offers a modulation/sweep unit. the former is similar to FDS' but has much reduced speed control.

## effects

---

- 10xx: **set waveform.**
- 11xx: **set noise length.** range is 0 to 7.
  - only in the noise channel.
- 12xy: **setup envelope.**
  - x determines whether envelope is enabled or not.
    - 0: disabled
    - 1: enabled
    - 3: enabled and loop
    - yeah, the value 2 isn't useful.
  - y sets the speed and direction.
    - 0-7: down
    - 8-F: up
- 13xy: **setup sweep.**
  - x sets the speed.
    - 0 and 8 are "speed 0" - sweep is ineffective.
  - y sets the shift (0 to 7).
    - 8 and higher will mute the channel.
  - only in channel 5.
- 14xy: **setup modulation.**
  - x determines whether it's enabled or not.
    - 0: disabled
    - 1: enabled
    - 3: enabled and loop
    - 2 isn't useful here either.
  - y sets the speed.
    - 0 and 8 are "speed 0" - modulation is ineffective.
    - no, you can't really do Yamaha FM using this.
  - only in channel 5.
- 15xx: **set modulation wave.**
  - xx points to a wavetable. range is 0 to FF.
  - this is an alternative to setting the modulation wave through the instrument.

## info

---

this chip uses the [Virtual Boy<sub>202</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Waveform storage mode:** selects how will waveforms be loaded.
- Static: loads the first 5 waves only.
- Dynamic: allows switching between more than 5 waves, but loading a new wave forces all channels off and back on again, usually with an audible click.
  - **Disable anti-phase-reset:** disables the workaround to a phase reset that occurs on all channels when the waveform of any channel is changed.
  - **I don't care about hardware:** ignores the 5-wave limitation and allows dynamic waveforms without channel resets or clicking. *note:* this is not hardware accurate!

# Konami VRC6

---

the most popular expansion chip to the Famicom's sound system.

the chip has 2 pulse wave channels and one sawtooth channel.

volume register is 4 bit for pulse wave and 6 bit for sawtooth, but sawtooth output is corrupted when volume register value is too high. because this register is actually an 8 bit accumulator, its output may wrap around.

for that reason, the sawtooth channel has its own instrument type. setting volume macro and/or pattern editor volume setting too high (above 42/2A) may distort the waveform.

pulse wave duty cycle is 8-level. it can be ignored and it has potential for DAC at this case: volume register in this mode is DAC output and it can be PCM playback through this mode.

Furnace supports this routine for PCM playback, but it consumes a lot of CPU time in real hardware (even if conjunction with VRC6's integrated IRQ timer).

## effects

---

these effects only are effective in the pulse channels.

- 12xx: **set duty cycle.** range is 0 to 7.
- 17xx: **toggle LEGACY sample mode.**
- **this effect exists only for compatibility reasons! its use is NOT recommended. use Sample type instruments instead.**

## info

---

this chip uses the [VRC6<sub>203</sub>](#) and [VRC6 \(saw\)<sub>203</sub>](#) instrument editors.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# Watara Supervision

---

a failed competitor of Game Boy, straight from Taiwan. released in 1992, it had a tilting screen, \$50 price tag, very fast 4 MHz 6502-like CPU, framebuffer graphics and sound capabilities similar to Game Boy.

these are 2 pulse wave channels (same as on GB), barely working PCM channel and noise channel, also not unlike Game Boy. no hardware envelopes or zombie mode, though.

## effects

---

- 12xx: **set duty cycle/noise mode.**
- range is 0 to 3 for pulse and 0 to 1 for noise.

## info

---

this chip uses the [Watara Supervision](#)<sup>204</sup> and [Generic Sample](#)<sup>183</sup> instrument editors.

## sample info

sample channel is a 4-bit DMA channel with 4 frequencies assigned to sample octaves (C2, C3, C4, C5; C, C#, D, D# in Furnace respectively). max sample size is 4 kilobytes (8192 samples).

## chip config

---

the following options are available in the Chip Manager window:

- **Swap noise duty cycles:** enabled by default. when enabled, short noise is on odd-indexed duty cycles, like on Game Boy, rather than even.
- **Stereo pulse waves:** disabled by default. when enabled, it forces pulse channel 1 to the right output channel and second pulse to the left output channel.

# WonderSwan

---

a handheld console released only in Japan by Bandai, designed by the same people behind Game Boy and Virtual Boy.

for this reason it has lots of similar elements from those two systems in the sound department.

it has 4 wavetable channels. some of them have additional capabilities:

- the second channel could play samples
- the third one has hardware sweep
- the fourth one also does noise

## effects

---

- 10xx: **change wave.**
- 11xx: **setup noise mode.** channel 4 only.
- 0: disable.
- 1-8: enable and set length.
- 12xx: **setup sweep period.** channel 3 only.
- 0: disable.
- 1-32: enable and set period.
- 13xx: **setup sweep amount.** channel 3 only.
- 00 to 7F for 0 to 127.
- 80 to FF for -128 to -1.
- 20xx: **set internal speaker loudness.**
- 0-1: 100% (default).
- 2-3: 200%.
- 4-7: 400%.
- 8: 800%.
- has no effect when "Headphone output" is on. see "chip config" below.

## info

---

this chip uses the [WonderSwan<sub>207</sub>](#) instrument editor.

## chip config

---

the following option is available in the Chip Manager window:

- **Headphone output:** enables stereo 16-bit output. if disabled, the internal speaker's 8-bit mono output is used. default is on.







## channel status

---

the following icons are displayed when channel status is enabled in the pattern view:

- PCM mode (channel 2):



-  off
-  on
- sweep (channel 3):
  -  disabled
  -  enabled
- noise mode (channel 4):
  -  off
  -  on

# Seta/Allumer X1-010

---

the X1-010 is a chip used by Seta (and Allumer) in the Seta 1 and 2 arcade boards.

it has 16 channels of wavetable sound with some support for 8-bit samples up to 128KB in length. the sample frequency resolution is pretty bad in the low end though...

even though this chip has stereo output, no board (as far as we know) uses the two outputs that it has... instead, only one output is connected, effectively being used as a mono chip.

the chip also has some (complicated) hardware volume envelope capabilities, with half of its memory being usable for that purpose. the shape of a volume envelope is defined by user-provided 128/16 waveforms.

the chip can store up to 32 sound waveforms and envelope waveforms at once.

this chip was the inspiration for Organya/PxTone (the former being used in a well-known game called Cave Story).

## effects

---

- 10xx: **change wave.**
- 11xx: **change envelope shape.** also wavetable.
- 17xx: **toggle LEGACY sample mode.**
- **this effect exists only for compatibility reasons! its use is NOT recommended. use Sample type instruments instead.**
- 20xx: **set PCM frequency.** range is 1 to FF.
- PCM frequency formula:  $\text{step} * (\text{clock} / 8192)$ .
- range is 1.95KHz to 498KHz if the chip clock is 16MHz.
- 22xx: **set envelope mode.**
- bit 0 sets whether envelope will affect this channel.
- bit 1 sets whether envelope will run once instead of looping.
- bit 2 sets whether split mode is used. I don't know what it does.
- bit 3/5 sets whether the right/left shape will mirror the original one.
- bit 4/6 sets whether the right/left output will mirror the original one.
- 23xx: **set envelope period.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**
- 29xy: **enable auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.

## info

---

this chip uses the [X1-010<sub>209</sub>](#) instrument editor.

# chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Stereo:** enables stereo output. the chip supports this, but none of the actual boards use it.
- **Bankswitched:** don't worry about it for now...

# Yamaha YM2151

---

the sound chip powering several arcade boards, synthesizers and the Sharp X1/X68000. eight 4-op FM channels, with overpowered LFO and almost unused noise generator.

it also was present on several pinball machines and synthesizers of the era, and later surpassed by the YM2414 (OPZ) present in the world-famous TX81Z.

in most arcade boards the chip was used in combination with a PCM chip, like [SegaPCM<sub>290</sub>](#) or [OKI's line of ADPCM chips<sub>258</sub>](#).

## effects

---

- 10xx: **set noise frequency of channel 8 operator 4.** 00 disables noise while 01 to 20 enables it.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value..
- 17xx: **set LFO speed.**
- 18xx: **set LFO waveform.**
- 00: saw
- 01: square
- 02: triangle
- 03: noise
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 1Exx: **set LFO AM depth.**
- 1Fxx: **set LFO PM depth.**
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 53xy: **set DT of operator.**

- x is the operator (1-4). a value of 0 means "all operators".
- y is the value:
  - 0: +0
  - 1: +1
  - 2: +2
  - 3: +3
  - 4: -0
  - 5: -3
  - 6: -2
  - 7: -1
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set DT2 of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxx: **set D2R/SR of all operators.**
- 5Cxx: **set D2R/SR of operator 1.**
- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 60xy: **set operator mask.**
- enables or disables operators.
- if x is 0, y ranges from 0 to F. it is a bitfield, so y is the sum of the active operators' bits:
  - OP1 is +1, OP2 is +2, OP3 is +4, and OP4 is +8.
  - for example, having only OP2 and OP4 on would be 2 + 8 = 10, resulting in an xy value of 0A.
- if x is 1 to 4, the effect targets that operator; y turns it off with a value of 0 and on with a value of 1.
  - for example, the effect 6031 enables OP3.
- 61xx: **set algorithm** (0 to 7).
- 62xx: **set LFO FM depth** (0 to 7).
- 63xx: **set LFO AM depth** (0 to 3).

## info

---

this chip uses the [FM \(OPM\)<sub>145</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Broken pitch macro/slides:** due to an oversight, pitch slides were twice as fast in older versions of Furnace. this option exists for compatibility.

# Yamaha YM2203 (OPN)

---

a cost-reduced version of the YM2151 (OPM).

it only has 3 FM channels instead of 8 and removes stereo, the LFO and DT2 (coarse detune).

however it does contain an AY/SSG part which provides 3 channels of square wave with noise and envelope.

this chip was used in the NEC PC-88/PC-98 series of computers, the Fujitsu FM-7AV and in some arcade boards.

several variants of this chip were released as well, with more features.

## effects

---

- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator from 1 to 4.
- y is the new MULT value..
- 18xx: **toggle extended channel 3 mode.**
- 0 disables it and 1 enables it.
- only in extended channel 3 chip.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 20xx: **set SSG channel mode.** xx may be one of the following:
  - 0: square
  - 1: noise
  - 2: square and noise
  - 3: nothing (apparently)
  - 4: envelope and square
  - 5: envelope and noise
  - 6: envelope and square and noise
  - 7: nothing
- 21xx: **set noise frequency.** xx is a value between 00 and 1F.
- 22xy: **set envelope mode.**
- x sets the envelope shape, which may be one of the following:
  - 0: \\_\_\_ decay
  - 4: /\_\_\_ attack once
  - 8: \\\ saw
  - 9: \\_\_\_ decay
  - A: \/\ inverse obelisco

- B: \ ^ ^ ^ decay once
- C: / / / / inverse saw
- D: / ^ ^ ^ attack
- E: / \ / \ obelisco
- F: / \_ \_ \_ attack once
- if y is 1 then the envelope will affect this channel.
- 23xx: **set envelope period low byte.**
- 24xx: **set envelope period high byte.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**
- 29xy: **enable SSG auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 53xy: **set DT of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value:
  - 0: +0
  - 1: +1
  - 2: +2
  - 3: +3
  - 4: -0
  - 5: -3
  - 6: -2
  - 7: -1
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set SSG-EG of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value (0-8).
  - values between 0 and 7 set SSG-EG.
  - value 8 disables it.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxx: **set D2R/SR of all operators.**

- 5Cxx: **set D2R/SR of operator 1.**
- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 60xy: **set operator mask.**
- enables or disables operators.
- if x is 0, y ranges from 0 to F. it is a bitfield, so y is the sum of the active operators' bits:
  - OP1 is +1, OP2 is +2, OP3 is +4, and OP4 is +8.
  - for example, having only OP2 and OP4 on would be 2 + 8 = 10, resulting in an xy value of 0A.
- if x is 1 to 4, the effect targets that operator; y turns it off with a value of 0 and on with a value of 1.
  - for example, the effect 6031 enables OP3.
- 61xx: **set algorithm** (0 to 7).

## info

---

this chip uses the [FM \(OPN\)<sub>148</sub>](#) and [AY-3-8910/SSG<sub>124</sub>](#) instrument editor.

## extended channel 3

---

in ExtCh mode, channel 3 is split into one column for each of its four operators and feedback are shared. the frequency of each operator may be controlled independently with notes and effects. this can be used for more polyphony or more complex sounds.

all four operators are still combined according to the algorithm in use. for example, algorithm 7 acts as four independent sine waves. algorithm 4 acts as two independent 2op sounds. even with algorithm 0, placing a note in any operator triggers that operator alone.

## SSG-EG

---

SSG-EG is short for "Software-controlled Sound Generator – Envelope Generator". it is the AY-3-8910/YM2149 envelope generator applied to each individual operator. it makes the operator's envelope play through attack, decay, sustain, and decay 2 until it reaches zero amplitude, at which time SSG triggers. according to the shape of the SSG envelope, the operator's envelope may then either loop or hold, and either of these can be set to invert the envelope (attack decreases and decay increases) when triggered.

a full guide to SSG-EG is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUnWU) (https://www.youtube.com/watch?v=IKOR0TUnWU) , this [detailed technical explanation](https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) (https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) , and this [chart of tunings](https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJFlod3FkQ3uq86rYy1VBluZc/) (https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJFlod3FkQ3uq86rYy1VBluZc/).

## CSM

---

CSM, or "Composite Sine Mode", involves a timer matching the frequency of the note in the "CSM Timer" channel. each time it triggers, it generates key-on and key-off commands to reset the phase of all operators on channel 3 and force their envelopes to restart at the release point. this can be



used to create vocal formants (speech synthesis!) or other complex effects. outside this chip's specific implementation, the technique is known as "oscillator sync".

working with CSM is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUlWU) (https://www.youtube.com/watch?v=IKOR0TUlWU) .

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Output rate:** sets the "prescale" parameter of the chip. allows you to run at a lower clock rate.
- **SSG Volume:** sets volume of SSG part.
- **FM Volume:** sets volume of FM part.

# Yamaha YM2608 (OPNA)

---

like YM2203, but with twice the FM channels, stereo, an ADPCM channel and built-in drums ("rhythm")!

it was one of the available sound chips for the NEC PC-88VA and later models of PC-98 series of computers.

the YM2610 (OPNB) and YM2610B chips are very similar to this one, but the built-in drums have been replaced with 6 sample channels.

## effects

---

- 10xy: **set LFO parameters.**
- x toggles the LFO.
- y sets its speed.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value..
- 18xx: **toggle extended channel 3 mode.**
- 0 disables it and 1 enables it.
- only in extended channel 3 chip.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 20xx: **set SSG channel mode.**
- 00: square
- 01: noise
- 02: square and noise
- 03: nothing (apparently)
- 04: envelope and square
- 05: envelope and noise
- 06: envelope and square and noise
- 07: nothing
- 21xx: **set noise frequency.** range is 0 to 1F.
- 22xy: **set envelope mode.**
- x sets the envelope shape, which may be one of the following:
  - 0: \\_\_\_ decay
  - 4: /\_\_\_ attack once
  - 8: \\\ saw
  - 9: \\_\_\_ decay

- A: \/\ / inverse obelisco
- B: \ --- decay once
- C: // // inverse saw
- D: / --- attack
- E: /\ / obelisco
- F: / \_\_\_ attack once
- if y is 1 then the envelope will affect this channel.
- 23xx: **set envelope period low byte.**
- 24xx: **set envelope period high byte.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**
- 29xy: **enable SSG auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 53xy: **set DT of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value:
  - 0: +0
  - 1: +1
  - 2: +2
  - 3: +3
  - 4: -0
  - 5: -3
  - 6: -2
  - 7: -1
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set SSG-EG of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value (0-8).
  - values between 0 and 7 set SSG-EG.
  - value 8 disables it.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**

- 5Bxx: **set D2R/SR of all operators.**
- 5Cxx: **set D2R/SR of operator 1.**
- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 60xy: **set operator mask.**
- enables or disables operators.
- if x is 0, y ranges from 0 to F. it is a bitfield, so y is the sum of the active operators' bits:
  - OP1 is +1, OP2 is +2, OP3 is +4, and OP4 is +8.
  - for example, having only OP2 and OP4 on would be 2 + 8 = 10, resulting in an xy value of 0A.
- if x is 1 to 4, the effect targets that operator; y turns it off with a value of 0 and on with a value of 1.
  - for example, the effect 6031 enables OP3.
- 61xx: **set algorithm** (0 to 7).
- 62xx: **set LFO FM depth** (0 to 7).
- 63xx: **set LFO AM depth** (0 to 3).

## info

---

this chip uses the [FM \(OPN\)](#)<sup>148</sup>, [AY-3-8910/SSG](#)<sup>124</sup>, [ADPCM-A](#)<sup>122</sup> and [ADPCM-B](#)<sup>123</sup> instrument editors.

## extended channel 3

---

in ExtCh mode, channel 3 is split into one column for each of its four operators. feedback and LFO levels are shared. the frequency of each operator may be controlled independently with notes and effects. this can be used for more polyphony or more complex sounds.

all four operators are still combined according to the algorithm in use. for example, algorithm 7 acts as four independent sine waves. algorithm 4 acts as two independent Zop sounds. even with algorithm 0, placing a note in any operator triggers that operator alone.

## SSG-EG

---

SSG-EG is short for "Software-controlled Sound Generator – Envelope Generator". it is the AY-3-8910/YM2149 envelope generator applied to each individual operator. it makes the operator's envelope play through attack, decay, sustain, and decay 2 until it reaches zero amplitude, at which time SSG triggers. according to the shape of the SSG envelope, the operator's envelope may then either loop or hold, and either of these can be set to invert the envelope (attack decreases and decay increases) when triggered.

a full guide to SSG-EG is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUnWU) (https://www.youtube.com/watch?v=IKOR0TUnWU) , this [detailed technical explanation](https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) (https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) , and this [chart of tunings](https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJfIdod3FkQ3uq86rYy1VBluZc/) (https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJfIdod3FkQ3uq86rYy1VBluZc/).

# CSM

---

CSM, or "Composite Sine Mode", involves a timer matching the frequency of the note in the "CSM Timer" channel. each time it triggers, it generates key-on and key-off commands to reset the phase of all operators on channel 3 and force their envelopes to restart at the release point. this can be used to create vocal formants (speech synthesis!) or other complex effects. outside this chip's specific implementation, the technique is known as "oscillator sync".

working with CSM is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUlWU) (<https://www.youtube.com/watch?v=IKOR0TUlWU>) .

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Output rate:** sets the "prescale" parameter of the chip. allows you to run at a lower clock rate.
- **SSG Volume:** sets volume of SSG part.
- **FM Volume:** sets volume of FM part.

# Neo Geo/Yamaha YM2610

---

originally an arcade board, but SNK shortly adapted it to a rather expensive video game console with the world's biggest cartridges because some people liked the system so much they wanted a home version of it.

its soundchip is a 4-in-1: 4ch 4-op FM, YM2149 (AY-3-8910 clone) and 2 different format ADPCM (<https://wiki.neogeodev.org/index.php?title=ADPCM>) in a single package!

## effects

---

- 10xy: **set LFO parameters.**
- x toggles the LFO.
- y sets its speed.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value..
- 18xx: **toggle extended channel 2 mode.**
- 0 disables it and 1 enables it.
- only in extended channel 2 chip.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 20xx: **set SSG channel mode.**
- 0: square
- 1: noise
- 2: square and noise
- 3: nothing (apparently)
- 4: envelope and square
- 5: envelope and noise
- 6: envelope and square and noise
- 7: nothing
- 21xx: **set noise frequency.** range is 00 to 1F.
- 22xy: **set envelope mode.**
- x sets the envelope shape:
  - 0: \\_\_\_ decay
  - 4: /\_\_\_ attack once
  - 8: \\\ saw
  - 9: \\_\_\_ decay
  - A: \/\ inverse obelisco
  - B: \ - - - decay once

- C: /// inverse saw
- D: /--- attack
- E: /\ /\ obelisco
- F: /\_\_\_ attack once
- if y is 1 then the envelope will affect this channel.
- 23xx: **set envelope period low byte.**
- 24xx: **set envelope period high byte.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**
- 29xy: **enable SSG auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 53xy: **set DT of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value:
  - 0: +0
  - 1: +1
  - 2: +2
  - 3: +3
  - 4: -0
  - 5: -3
  - 6: -2
  - 7: -1
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set SSG-EG of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value (0-8).
  - values between 0 and 7 set SSG-EG.
  - value 8 disables it.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxx: **set D2R/SR of all operators.**
- 5Cxx: **set D2R/SR of operator 1.**

- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 60xy: **set operator mask.**
- enables or disables operators.
- if x is 0, y ranges from 0 to F. it is a bitfield, so y is the sum of the active operators' bits:
  - OP1 is +1, OP2 is +2, OP3 is +4, and OP4 is +8.
  - for example, having only OP2 and OP4 on would be 2 + 8 = 10, resulting in an xy value of 0A.
- if x is 1 to 4, the effect targets that operator; y turns it off with a value of 0 and on with a value of 1.
  - for example, the effect 6031 enables OP3.
- 61xx: **set algorithm** (0 to 7).
- 62xx: **set LFO FM depth** (0 to 7).
- 63xx: **set LFO AM depth** (0 to 3).

## info

---

this chip uses the [FM \(OPN\)](#)<sup>148</sup>, [AY-3-8910/SSG](#)<sup>124</sup>, [ADPCM-A](#)<sup>122</sup> and [ADPCM-B](#)<sup>123</sup> instrument editors.

## extended channel 2

---

in ExtCh mode, channel 2 is split into one column for each of its four operators. feedback and LFO levels are shared. the frequency of each operator may be controlled independently with notes and effects. this can be used for more polyphony or more complex sounds.

all four operators are still combined according to the algorithm in use. for example, algorithm 7 acts as four independent sine waves. algorithm 4 acts as two independent 2op sounds. even with algorithm 0, placing a note in any operator triggers that operator alone.

## SSG-EG

---

SSG-EG is short for "Software-controlled Sound Generator – Envelope Generator". it is the AY-3-8910/YM2149 envelope generator applied to each individual operator. it makes the operator's envelope play through attack, decay, sustain, and decay 2 until it reaches zero amplitude, at which time SSG triggers. according to the shape of the SSG envelope, the operator's envelope may then either loop or hold, and either of these can be set to invert the envelope (attack decreases and decay increases) when triggered.

a full guide to SSG-EG is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUlnWU) (https://www.youtube.com/watch?v=IKOR0TUlnWU) , this [detailed technical explanation](https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) (https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) , and this [chart of tunings](https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJfIdod3FkQ3uq86rYy1VBluZc/) (https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJfIdod3FkQ3uq86rYy1VBluZc/).

## CSM

---

CSM, or "Composite Sine Mode", involves a timer matching the frequency of the note in the "CSM Timer" channel. each time it triggers, it generates key-on and key-off commands to reset the phase of all operators on channel 2 and force their envelopes to restart at the release point. this can be



used to create vocal formants (speech synthesis!) or other complex effects. outside this chip's specific implementation, the technique is known as "oscillator sync".

working with CSM is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUlWU) (https://www.youtube.com/watch?v=IKOR0TUlWU) .

## chip config

---

the following options are available in the Chip Manager window:

- sets the rate at which the chip will run.
- **SSG Volume**: sets volume of SSG part.
- **FM Volume**: sets volume of FM part.

# Taito Arcade/Yamaha

## YM2610B

---

YM2610B is basically YM2610 with 2 extra FM channels used at some 90s Taito arcade hardware. it is backward compatible with the original chip.

### effects

---

- 10xy: **set LFO parameters.**
- x toggles the LFO.
- y sets its speed.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value..
- 18xx: **toggle extended channel 3 mode.**
- 0 disables it and 1 enables it.
- only in extended channel 3 chip.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 20xx: **set SSG channel mode.**
- 00: square
- 01: noise
- 02: square and noise
- 03: nothing (apparently)
- 04: envelope and square
- 05: envelope and noise
- 06: envelope and square and noise
- 07: nothing
- 21xx: **set noise frequency.** range is 00 to 1F.
- 22xy: **set envelope mode.**
- x sets the envelope shape, which may be one of the following:
  - 0:\\_\_\_ decay
  - 4:/\_\_\_ attack once
  - 8:\\\\ saw
  - 9:\\_\_\_ decay
  - A:\// inverse obelisco
  - B:\^-^- decay once

- C://// inverse saw
- D:/-/-/- attack
- E:/\/\ obelisco
- F:/\_\_\_ attack once
- if y is 1 then the envelope will affect this channel.
- 23xx: **set envelope period low byte.**
- 24xx: **set envelope period high byte.**
- 25xx: **slide envelope period up.**
- 26xx: **slide envelope period down.**
- 29xy: **enable SSG auto-envelope mode.**
- in this mode the envelope period is set to the channel's notes, multiplied by a fraction.
- x is the numerator.
- y is the denominator.
- if x or y are 0 this will disable auto-envelope mode.
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 53xy: **set DT of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value:
  - 0: +0
  - 1: +1
  - 2: +2
  - 3: +3
  - 4: -0
  - 5: -3
  - 6: -2
  - 7: -1
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set SSG-EG of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value (0-8).
  - values between 0 and 7 set SSG-EG.
  - value 8 disables it.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxx: **set D2R/SR of all operators.**
- 5Cxx: **set D2R/SR of operator 1.**

- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 60xy: **set operator mask.**
- enables or disables operators.
- if x is 0, y ranges from 0 to F. it is a bitfield, so y is the sum of the active operators' bits:
  - OP1 is +1, OP2 is +2, OP3 is +4, and OP4 is +8.
  - for example, having only OP2 and OP4 on would be 2 + 8 = 10, resulting in an xy value of 0A.
- if x is 1 to 4, the effect targets that operator; y turns it off with a value of 0 and on with a value of 1.
  - for example, the effect 6031 enables OP3.
- 61xx: **set algorithm** (0 to 7).
- 62xx: **set LFO FM depth** (0 to 7).
- 63xx: **set LFO AM depth** (0 to 3).

## info

---

this chip uses the [FM \(OPN\)](#)<sup>148</sup>, [AY-3-8910/SSG](#)<sup>124</sup>, [ADPCM-A](#)<sup>122</sup> and [ADPCM-B](#)<sup>123</sup> instrument editors.

## extended channel 3

---

in ExtCh mode, channel 3 is split into one column for each of its four operators. feedback and LFO levels are shared. the frequency of each operator may be controlled independently with notes and effects. this can be used for more polyphony or more complex sounds.

all four operators are still combined according to the algorithm in use. for example, algorithm 7 acts as four independent sine waves. algorithm 4 acts as two independent 2op sounds. even with algorithm 0, placing a note in any operator triggers that operator alone.

## SSG-EG

---

SSG-EG is short for "Software-controlled Sound Generator – Envelope Generator". it is the AY-3-8910/YM2149 envelope generator applied to each individual operator. it makes the operator's envelope play through attack, decay, sustain, and decay 2 until it reaches zero amplitude, at which time SSG triggers. according to the shape of the SSG envelope, the operator's envelope may then either loop or hold, and either of these can be set to invert the envelope (attack decreases and decay increases) when triggered.

a full guide to SSG-EG is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUlnWU) (https://www.youtube.com/watch?v=IKOR0TUlnWU) , this [detailed technical explanation](https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) (https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) , and this [chart of tunings](https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJfIdod3FkQ3uq86rYy1VBluZc/) (https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJfIdod3FkQ3uq86rYy1VBluZc/).

## CSM

---

CSM, or "Composite Sine Mode", involves a timer matching the frequency of the note in the "CSM Timer" channel. each time it triggers, it generates key-on and key-off commands to reset the phase of all operators on channel 3 and force their envelopes to restart at the release point. this can be

used to create vocal formants (speech synthesis!) or other complex effects. outside this chip's specific implementation, the technique is known as "oscillator sync".

working with CSM is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TUlWU) (<https://www.youtube.com/watch?v=IKOR0TUlWU>) .

## chip config

---

the following options are available in the Chip Manager window:

- sets the rate at which the chip will run.
- **SSG Volume**: sets volume of SSG part.
- **FM Volume**: sets volume of FM part.

# Yamaha YM2612

---

one of two chips that powered the Sega Genesis. it is a six-channel, four-operator FM synthesizer. channel 6 can be turned into 8-bit PCM player, that via software mixing, thanks to Z80 sound CPU, can play more than one channel of straight-shot samples at once.

Furnace also offers DualPCM, a Z80 driver that splits channel 6 into two individual PCM channels with variable pitch. using the console's Z80 processor, these are mixed together in software and streamed to channel 6 in PCM mode with a mix rate of 13750 Hz. VGM export requires the "direct stream mode" option to be enabled, and resulting files will be very large.

## effects

---

- 10xy: **set LFO parameters.**
- x toggles the LFO.
- y sets its speed.
- 11xx: **set feedback of channel.**
- 12xx: **set operator 1 level.**
- 13xx: **set operator 2 level.**
- 14xx: **set operator 3 level.**
- 15xx: **set operator 4 level.**
- 16xy: **set multiplier of operator.**
- x is the operator (1-4).
- y is the new MULT value.
- 17xx: **toggle LEGACY sample mode.**
- this only works on channel 6.
- **this effect exists only for compatibility reasons! its use is NOT recommended. use Sample type instruments instead.**
- 18xx: **toggle extended channel 3 mode.**
- 0 disables it and 1 enables it.
- only in extended channel 3 chip.
- 19xx: **set attack of all operators.**
- 1Axx: **set attack of operator 1.**
- 1Bxx: **set attack of operator 2.**
- 1Cxx: **set attack of operator 3.**
- 1Dxx: **set attack of operator 4.**
- 30xx: **enable envelope hard reset.**
- this works by inserting a quick release and tiny delay before a new note.
- 50xy: **set AM of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy: **set SL of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 52xy: **set RR of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 53xy: **set DT of operator.**

- x is the operator (1-4). a value of 0 means "all operators".
- y is the value:
  - 0: -3
  - 1: -2
  - 2: -1
  - 3: 0
  - 4: 1
  - 5: 2
  - 6: 3
  - 7: -0
- 54xy: **set RS of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value.
- 55xy: **set SSG-EG of operator.**
- x is the operator (1-4). a value of 0 means "all operators".
- y is the value (0-8).
  - values between 0 and 7 set SSG-EG.
  - value 8 disables it.
- 56xx: **set DR of all operators.**
- 57xx: **set DR of operator 1.**
- 58xx: **set DR of operator 2.**
- 59xx: **set DR of operator 3.**
- 5Axx: **set DR of operator 4.**
- 5Bxx: **set D2R/SR of all operators.**
- 5Cxx: **set D2R/SR of operator 1.**
- 5Dxx: **set D2R/SR of operator 2.**
- 5Exx: **set D2R/SR of operator 3.**
- 5Fxx: **set D2R/SR of operator 4.**
- 60xy: **set operator mask.**
- enables or disables operators.
- if x is 0, y ranges from 0 to F. it is a bitfield, so y is the sum of the active operators' bits:
  - OP1 is +1, OP2 is +2, OP3 is +4, and OP4 is +8.
  - for example, having only OP2 and OP4 on would be 2 + 8 = 10, resulting in an xy value of 0A.
- if x is 1 to 4, the effect targets that operator; y turns it off with a value of 0 and on with a value of 1.
  - for example, the effect 6031 enables OP3.
- 61xx: **set algorithm** (0 to 7).
- 62xx: **set LFO FM depth** (0 to 7).
- 63xx: **set LFO AM depth** (0 to 3).

## info

---

this chip uses the [FM \(OPN\)](#)<sup>148</sup> and [Generic Sample](#)<sup>183</sup> instrument editors.

## extended channel 3

---

in ExtCh mode, channel 3 is split into one column for each of its four operators. feedback and LFO levels are shared. the frequency of each operator may be controlled independently with notes and effects. this can be used for more polyphony or more complex sounds.

all four operators are still combined according to the algorithm in use. for example, algorithm 7 acts as four independent sine waves. algorithm 4 acts as two independent Zop sounds. even with algorithm 0, placing a note in any operator triggers that operator alone.

## CSM

---

CSM, or "Composite Sine Mode", involves a timer matching the frequency of the note in the "CSM Timer" channel. each time it triggers, it generates key-on and key-off commands to reset the phase of all operators on channel 3 and force their envelopes to restart at the release point. this can be used to create vocal formants (speech synthesis!) or other complex effects. outside this chip's specific implementation, the technique is known as "oscillator sync".

working with CSM is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TULnWU) (https://www.youtube.com/watch?v=IKOR0TULnWU) .

## DualPCM

---

thanks to the Z80 sound CPU, DualPCM can play two samples at once! this mode splits channel 6 into two individual PCM channels with variable pitch. these are mixed together in software and streamed to channel 6 with a mix rate of 13750 Hz. VGM export requires the "direct stream mode" option to be enabled, and resulting files will be very large.

## SSG-EG

---

SSG-EG is short for "Software-controlled Sound Generator – Envelope Generator". it is the AY-3-8910/YM2149 envelope generator applied to each individual operator. it makes the operator's envelope play through attack, decay, sustain, and decay 2 until it reaches zero amplitude, at which time SSG triggers. according to the shape of the SSG envelope, the operator's envelope may then either loop or hold, and either of these can be set to invert the envelope (attack decreases and decay increases) when triggered.

a full guide to SSG-EG is beyond the scope of this documentation. for more information, see this [brief SSG-EG and CSM video tutorial](https://www.youtube.com/watch?v=IKOR0TULnWU) (https://www.youtube.com/watch?v=IKOR0TULnWU) , this [detailed technical explanation](https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) (https://gendev.spritesmind.net/forum/viewtopic.php?t=386&start=106) , and this [chart of tunings](https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJFIdod3FkQ3uq86rYy1VBluZc/) (https://docs.google.com/spreadsheets/d/1HGKQ08CnLGAjA1U0StJFIdod3FkQ3uq86rYy1VBluZc/) .

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.
- **Chip type:** sets the chip type.
- YM3438: found in Model 2 and 3 Genesis. integrated version which fixes DAC distortion present in YM2612.
- YM2612: found in Model 1 Genesis. has DAC distortion.
- YMF276: found in FM Towns. uses external DAC and has higher output resolution.
- **DAC interrupt simulation:** simulates the short interrupts that occur on each frame with quirky drivers (e.g. as found in Sonic 2 and 3).



- not supported in VGM export!

# Yamaha YMU759

---

the Yamaha YMU759 is a sound chip designed for feature phones during the early 2000s. it is also known as MA-2.

sadly Yamaha didn't care about these chips too much, and the register specs were completely unavailable, which means the YMU759 is totally unsupported and unemulated besides Yamaha's official emulator for it built into MidRadio.

Furnace is able to load DefleMask modules written for this system; however, it doesn't support any of its effects and is simulated using the OPL core.

## effects

---

since this chip is so abandoned, there isn't any support for it.

even DefleMask dropped support for the chip in version 0.11, which just shows how poorly the chip has been treated, both by Yamaha themselves and the community.

hey, at least it was the spark that ignited the idea of DefleMask.

## info

---

this chip uses the [FM \(OPL\)](#)<sup>139</sup> and [Generic Sample](#)<sup>183</sup> instrument editors.

# Yamaha YMZ280B (PCMD8)

---

8-channel PCM/ADPCM sample-based sound chip designed for use with arcade machines. it lived throughout mid to late 90s.

it has 16-level stereo panning, up to 16-bit PCM and up to 16MB of external PCM data.

## effects

---

none so far.

## info

---

this chip uses the [YMZ280B<sub>210</sub>](#) instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **Clock rate:** sets the rate at which the chip will run.

# ZX Spectrum beeper

---

rather than having a dedicated sound synthesizer, early ZX Spectrum models had one piezo beeper, controlled by Z80 CPU and ULA chip. its capabilities should be on par with an IBM PC speaker... right?

not really - very soon talented programmers found out ways to output much more than one square wave channel. a lot of ZX beeper routines do exist, but Furnace only supports two engines:

- a Follin/SFX-like engine with 6 channels of narrow pulse wave and click drums.
- QuadTone: PWM-driven engine with 4ch of pulse wave with freely variable duty cycles and 1-bit PCM drums.

## effects

---

- 12xx: set pulse width.
- 17xx: trigger overlay drum.
- xx is the sample number.
- overlay drums are 1-bit and always play at 55930Hz (NTSC) or 55420Hz (PAL).
- the maximum length is 2048!

## info

---

this chip uses the [Beeper](#)<sup>126</sup> or [Pokémon Mini/QuadTone](#)<sup>174</sup> instrument editor.

## chip config

---

the following options are available in the Chip Manager window:

- **PAL**: run at 3.54MHz instead of 3.58MHz.

QuadTone provides this additional option:

- **Disable hissing**: disables TDM hissing.

# advanced

---

advanced Furnace features.

as listed in the "Edit" menu:

- [find/replace](#)<sup>368</sup>

as listed in the "Window" menu:

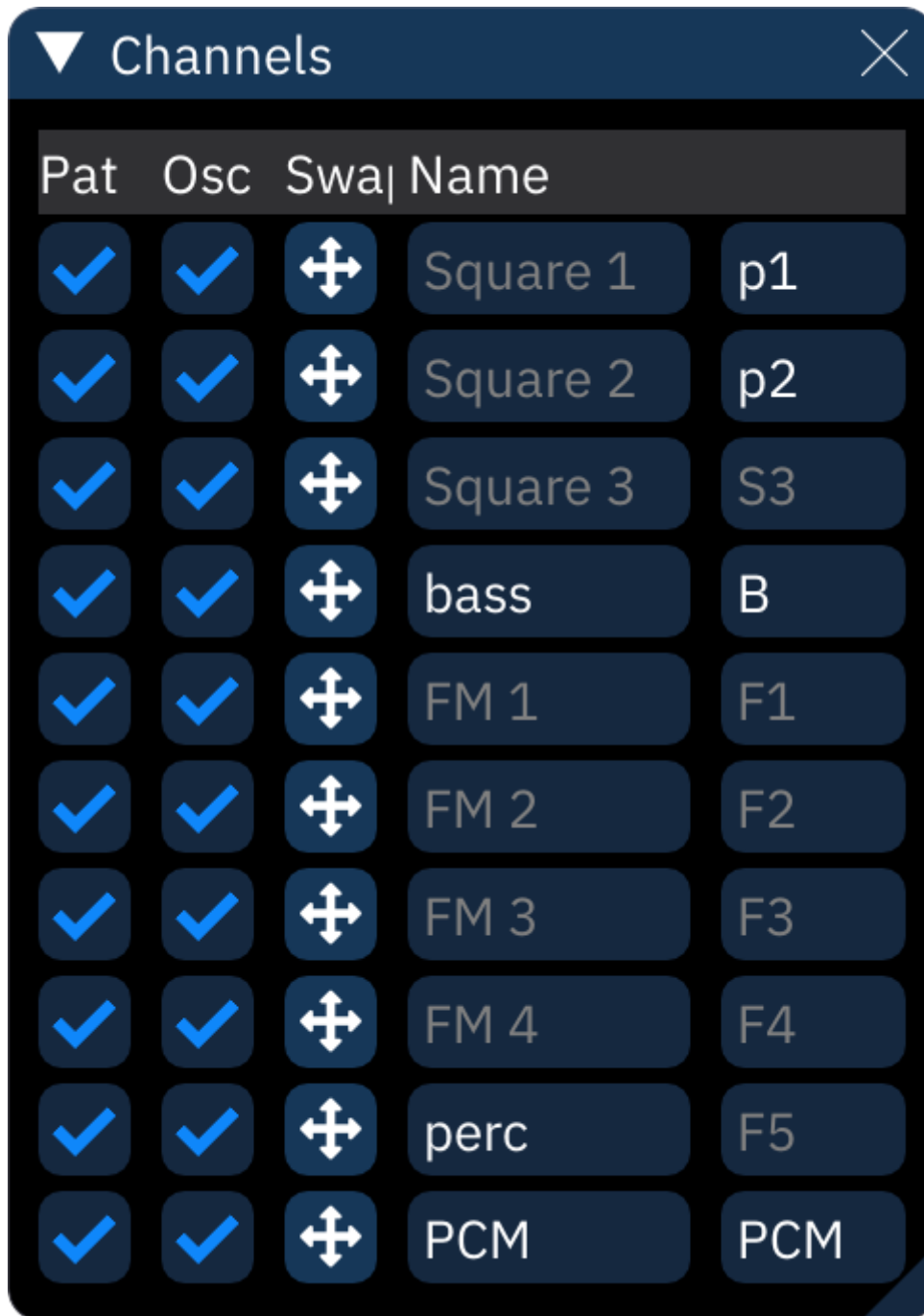
- song
- [song comments](#)<sup>366</sup>
- [channels](#)<sup>354</sup>
- [chip manager](#)<sup>360</sup>
- [pattern manager](#)<sup>382</sup>
- [mixer](#)<sup>378</sup>
- [compatibility flags](#)<sup>367</sup>
- visualizers
- [oscilloscope](#)<sup>381</sup>
- [oscilloscope \(per-channel\)](#)<sup>356</sup>
- [oscilloscope \(X-Y\)](#)<sup>390</sup>
- volume meter
- tempo
- [clock](#)<sup>362</sup>
- [grooves](#)<sup>372</sup>
- debug
- [log viewer](#)<sup>376</sup>
- [register view](#)<sup>386</sup>
- [statistics](#)<sup>387</sup>
- [memory composition](#)<sup>377</sup>
- [piano/input pad](#)<sup>384</sup>

other:

- [command line usage](#)<sup>363</sup>

# channels

the "Channels" dialog allows manipulation of the song's channels.



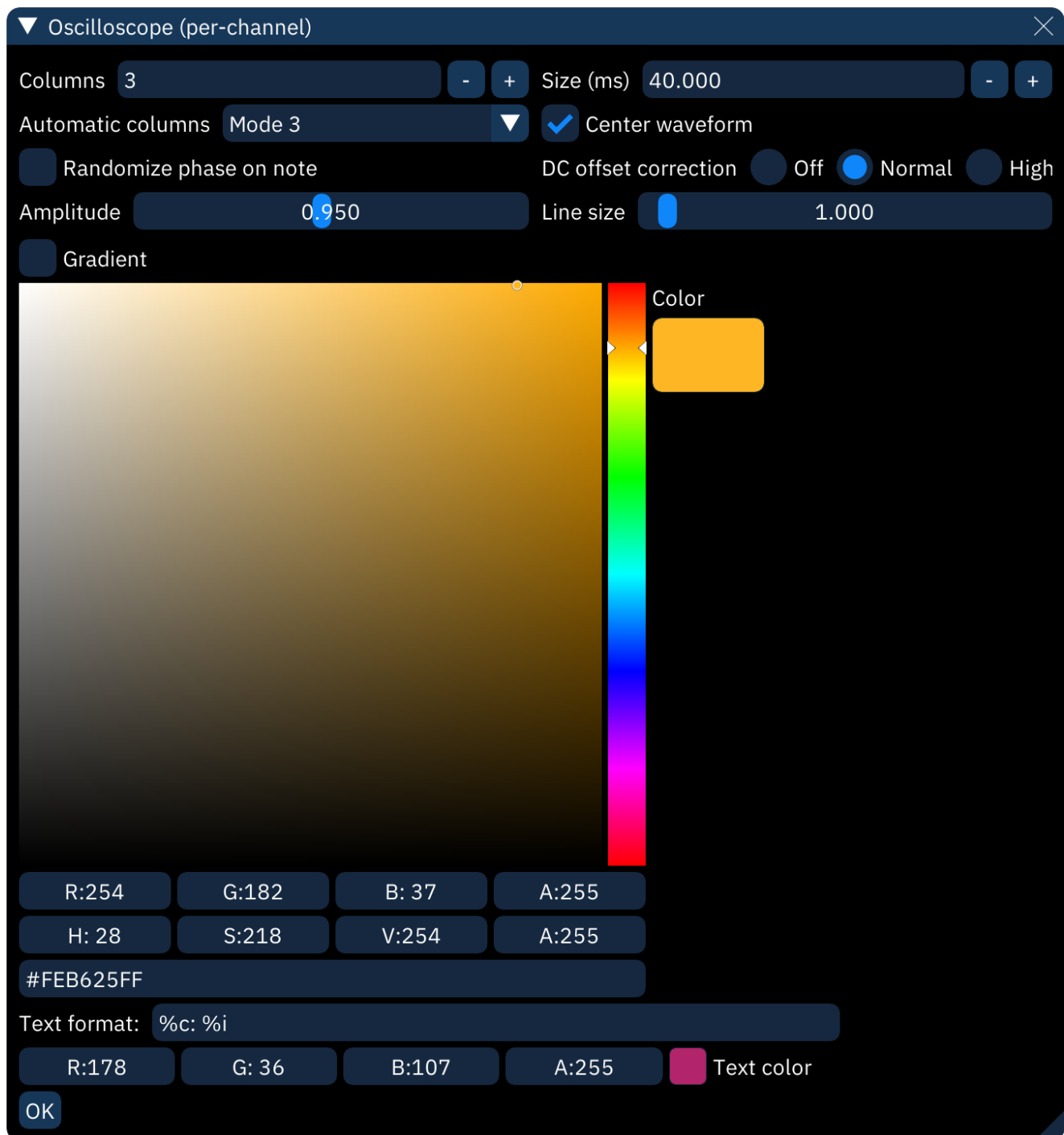
each channel has the following options:

- **Pat**: uncheck the box to hide the channel from the pattern view. pattern data will be kept.
- **Osc**: uncheck the box to hide the channel from the per-channel oscilloscope view.
- **Swap**: click and drag to rearrange pattern data throughout the song.
- note: this does **not** move channels around! it only moves the channel's pattern data.
- **Name**: the name displayed at the top of each channel in the pattern view.

- the next setting is "short name", which is displayed in the orders view and/or when a channel is collapsed.

# oscilloscope (per-channel)

the "Oscilloscope (per-channel)" window displays several oscilloscope views (one per channel).



right-clicking the view will display the configuration view shown above:

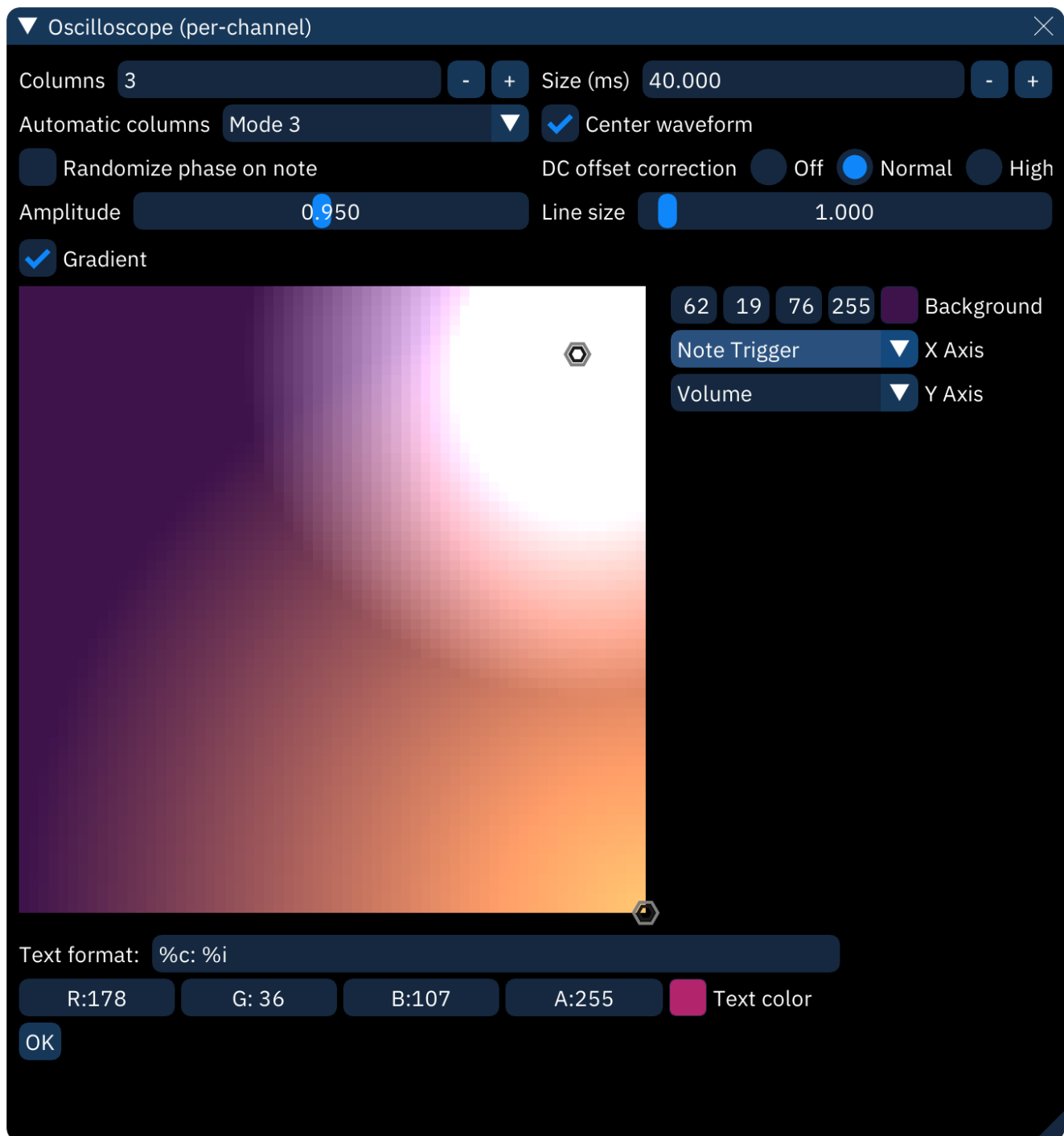
- **Columns:** sets the amount of columns for the oscilloscope views.
- **Size (ms):** sets how much of a channel's output is captured for the oscilloscope view.
- **Automatic columns:** sets the number of columns based on the number of channels.
- **Off:** use the Columns setting.



- **Mode 1:** always fewer columns than rows.
- **Mode 2:** bias slightly toward more columns.
- **Mode 3:** always more columns than rows.
- **Center waveform:** when enabled, the displayed waveforms will be centered horizontally using an auto-correlation algorithm.
- **Randomize phase on note**
- **DC offset correction:** controls the algorithm that centers the waveform vertically.
- **Off**
- **Normal**
- **High**
- **Amplitude:** scales amplitude for all oscilloscope views.
- **Line size:** controls line thickness.
- **Gradient:** this allows you to use a gradient for determining the waveforms' colors instead of a single color. see the gradient section for more information.
- if this option is off, a color selector will be displayed. right-click on it for some options:
  - select between the square selector and the color wheel selector.
  - **Alpha bar:** display an opacity bar.
- **Text format:** this allows you to display some text on each oscilloscope view. the following codes may be used:
  - %c: channel name
  - %C: channel short name
  - %d: channel number (starting from 0)
  - %D: channel number (starting from 1)
  - %n: channel note
  - %i: instrument name
  - %I: instrument number (decimal)
  - %x: instrument number (hex)
  - %s: chip name
  - %p: chip part number
  - %S: chip ID
  - %v: volume (decimal)
  - %V: volume (percentage)
  - %b: volume (hex)
  - %l: new line
  - %%: percent sign

click on OK to return to the main view.

# gradient



when enabling the Gradient setting, a gradient view is displayed in where circular "points" can be placed.

each point adds a color spot.

the resulting image is used to look up the waveform's color as determined by each axis.

- right-click to place a point.
- left-click on a point to change its color:
- a color picker is displayed, alongside two settings.
- **Distance**: the size of the circle.

- **Spread:** the size of the solid center of the circle. increasing it fills more of the circle with the color.
- middle-click on a point to delete it.
- **Background:** sets the gradient's background color.
- **X Axis:** determines what the horizontal axis maps to.
- **Y Axis:** determines what the vertical axis maps to. these can be set to the following:
- **None (0%):** always left or bottom
- **None (50%):** always center
- **None (100%):** always right or top
- **Frequency:** changes color with frequency
- **Volume:** changes color with volume
- **Channel:** changes color based on channel number (first channel is 0% and last is 100%)
- **Brightness:** currently does nothing
- **Note Trigger:** changes color when a new note is played

# chip manager

the **chip manager** window allows you to manage chips, including adding, changing, moving or re moving them.

▼ Chip Manager

☐ Preserve channel order

☒ Clone channel data

☐ Clone at end

Name	Actions
<div><div>⬆ ⬆</div>▶ 1. TI SN76489</div>	<div>Clone</div> <div>Change</div> <div>✕</div>
<div><div>⬆ ⬆</div>▼ 2. Yamaha YM2612 (OPN2)</div> <div>Clock rate:<div><div>●</div> NTSC (7.67MHz)</div><div><div>●</div> PAL (7.61MHz)</div><div><div>●</div> FM Towns (8MHz)</div><div><div>●</div> AtGames Genesis (6.13MHz)</div><div><div>●</div> Sega System 32 (8.05MHz)</div></div> <div>Chip type:<div><div>●</div> YM3438 (9-bit DAC)</div><div><div>●</div> YM2612 (9-bit DAC with distortion)</div><div><div>●</div> YMF276 (external DAC)</div></div> <div><div><input type="checkbox"/> Custom clock rate</div><div><div>0</div><div>-</div><div>+</div> Hz</div></div>	<div>Clone</div> <div>Change</div> <div>✕</div>

+

**Preserve channel order:** make existing pattern data stay in place even when chips are rearranged. when turned off, pattern data will be arranged to match (the default, and usually desired behavior).

**Clone channel data:** when cloning chips, also copy patterns, pattern names, channel names and other parameters to the clone.

**Clone at end:** instead of inserting the clone directly after the cloned chip, add it to the end.



to move a chip around, click and drag the button to the left.

to duplicate a chip, click the **Clone** button.

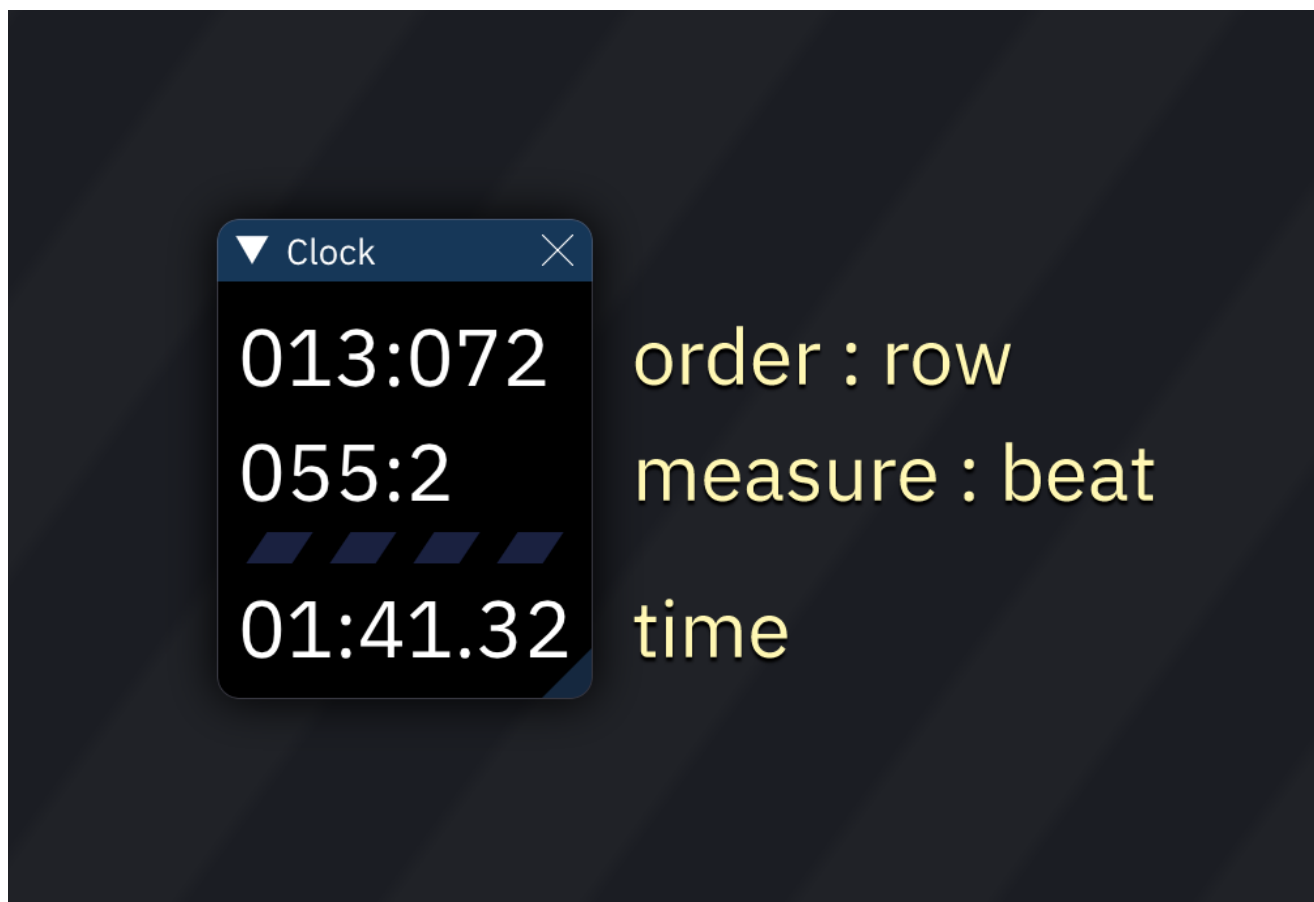
to replace a chip with a different one, click the **Change** button. this will display the chip selector.



to remove a chip, click the button.

click on a chip's name to open chip configuration. this allows you to change chip options, such as clock rate, chip type and so on.

# clock



the clock shows the current playback position relative to the start of the song:

- order : row
- measure : beat (as defined by row highlight settings)
- beat
- elapsed time in minutes:seconds.hundredths

# command line usage

---

## NAME

---

Furnace - a chiptune tracker

## SYNOPSIS

---

furnace [params...] [file]

## DESCRIPTION

---

Furnace is a chiptune tracker that supports many systems and sound chips from the 8/16-bit era. even though it is primarily controlled by using its graphical user interface, Furnace also offers a command line interface, which is described here.

## USAGE

---

starting Furnace without arguments will start the graphical user interface (GUI), as long as Furnace has been compiled with GUI enabled.

passing the path to a file will open that file at start-up. if Furnace cannot open that file, it will report an error and quit.

the following parameters may be used:

### general

- -help: display the following help.
- -console: enable command-line interface (CLI) player.
- see the COMMAND LINE INTERFACE section for more information
- -loglevel <level>: set the logging level to one of the following:
- error: critical errors only
- warning: errors and warnings
- info: errors, warnings, and useful information
- debug: all of the above, including debug information
- trace: like debug, but with even more details (default)
- -info: get information about a song.
- you must provide a file, otherwise Furnace will quit.
- -version: display version information.
- -warranty: view warranty disclaimer.

### engine

- -audio sdl|jack|portaudio: override audio backend to one of the following:
- sdl: SDL (default)
- jack: JACK Audio Connection Kit

- portaudio: PortAudio
- -view <type>: set visualization of data to one of the following:
- pattern: order and pattern
- commands: engine commands
- nothing: guess (default)
- -loops <count>: set number of loops
- -1 means loop forever.
- -subsong <number>: set sub-song to play.
- -safemode: enable safe mode (software rendering without audio).
- -safeaudio: enable safe mode (software rendering with audio).
- -benchmark render|seek: run performance test and output total time.
- render: measure render time
- seek: measure time to seek through the entire song
- you must provide a file, otherwise Furnace will quit.

### audio export

- -output path: export audio in .wav format to path.
- you must provide a file, otherwise Furnace will quit.
- -outmode one|persys|perchan: set audio export output mode.
- one: single file (default)
- persys: one file per chip (\_sXX will be appended to file name, where XX is the chip number)
- perchan: one file per channel (\_cXX will be appended to file name, where XX is the channel number)

### VGM export

- -vgmout path: output VGM data to path.
- you must provide a file, otherwise Furnace will quit.
- -direct: enable VGM export direct stream mode.
- this mode is useful for DualPCM export.
- note that this will increase file size by a huge amount!

### export (other)

- -cmdout path: output command stream dump to path.
- you must provide a file, otherwise Furnace will quit.
- -romout path: output ROM file export to path.
- you must provide a file, otherwise Furnace will quit.
- there must be an available ROM export target for the system.
- -romconf key=value: set a configuration parameter for -romout.
- you may use this multiple times to set multiple parameters.
- Amiga Validation
  - no parameters.
- Commander X16 ZSM
  - zsmrate: tick rate (Hz), default: 60
  - loop: loop song, default: true
  - optimize: optimize size, default: true
- Atari 2600 (TlunA)
  - baseLabel: base song label name, default: song
  - firstBankSize: max size in first bank, default: 3072
  - otherBankSize: max size in other banks, default: 4048



- sysToExport: TIA chip index, default: -1 (find first)
- Atari 8-bit SAP-R
  - no parameters.
- -txtout path: output text file export to path.
- you must provide a file, otherwise Furnace will quit.

## COMMAND LINE INTERFACE

---

Furnace provides a command-line interface (CLI) player which may be activated through the -console option.

the following controls may be used:

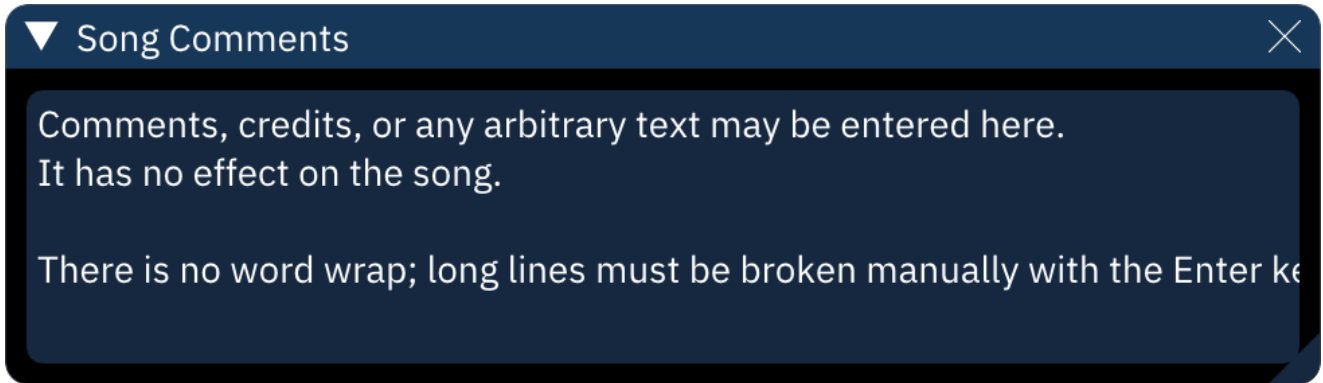
- Left/H: go to previous order.
- Right/L: go to next order.
- Space: pause/resume playback.

## SEE ALSO

---

the Furnace user manual in the manual.pdf file.

# comments



comments, credits, or any arbitrary text may be entered here. it has no effect on the song.

there is no word wrap; long lines must be broken manually with the Enter/Return key.

# compatibility flags

---

the **Compatibility Flags** window contains several tabs full of settings that change playback behavior. a new Furnace song will have these disabled, while opening a DefleMask module, .mod, or earlier Furnace file will automatically set the appropriate options.

hovering over most options will bring up additional info about them. it is not recommended to change any of these, especially the ones in the DefleMask and Old Furnace sections.

# find/replace

Furnace has a powerful find-and-replace function that can take the repetitive work out of mass editing.

## find

Find/Replace

Find

Replace

Note

between

C

3

G#

4

Ins

equals

13

Volume

ignore

Effect

equals

0A

Value

ignore

Add effect

Remove effect

Search range:

☒ Confine to channels

Match effect position:

Song

5: FM 5

From

Selection

8: Square 2

To

☒ Pattern

☒ No

☐ Lax

☐ Strict

Find

all data that can be found within a pattern can be searched for here.

a query contains:

- **Note:** note.
- **Ins:** instrument.
- **Volume:** volume.
- **Effect:** effect type.
- **Value:** effect value.

all of these have the following choices for what data will be found:

- **ignore**: ignore this.
- **equals**: match the given value exactly.
- **not equal**: match everything but the given value.
- **between**: match anything between and including the given values.
- **not between**: match anything outside the given range of values.
- **any**: match all values.
- **none**: match blanks only.

the following options also are available:

- -: remove query. if only one query exists, it is cleared.
- **Add effect**: adds another Effect and Value to the query, each set representing additional effects columns.
- **Remove effect**: removes last Effect and Value from the query.
- +: adds another query.
- **Search range**: restricts search range to the whole **Song**, the current **Selection**, or the currently viewed **Pattern**.
- **Confine to channels**: restricts to the selected channels and the channels between them.
- **Match effect position**: chooses how the order of effect types and effect values will matter when finding them.
- **No**: no attention is paid to what order the effects appear in.
- **Lax**: matches effects if they appear in the same order as selected above.
- **Strict**: effects may only match in their corresponding effects columns.
- **Find**: finds everything that matches the query and displays it in a list.
- the **order**, **row**, and **channel** columns are as they say.
- the **go** column of buttons will take you to the location of the result.

# replace

Find/Replace

Find

Replace

<input type="checkbox"/>	Note	set	▼	C-0	▼		
<input type="checkbox"/>	Ins	set	▼	00		-	+
<input type="checkbox"/>	Volume	set	▼	00		-	+
<input checked="" type="checkbox"/>	Effect	set	▼	F3		-	+
<input checked="" type="checkbox"/>	Value	scale %	▼	125		-	+

Add effect

Remove effect

Effect replace mode:

☐ Replace matches only

☒ Replace matches, then free spaces

☐ Clear effects

☐ Insert in free spaces

Replace

you may select any of these to replace:

- **Note:** note.
- **Ins:** instrument.
- **Volume:** volume.
- **Effect:** effect type.
- **Value:** effect value.

all of these have the following choices for how they alter matches:

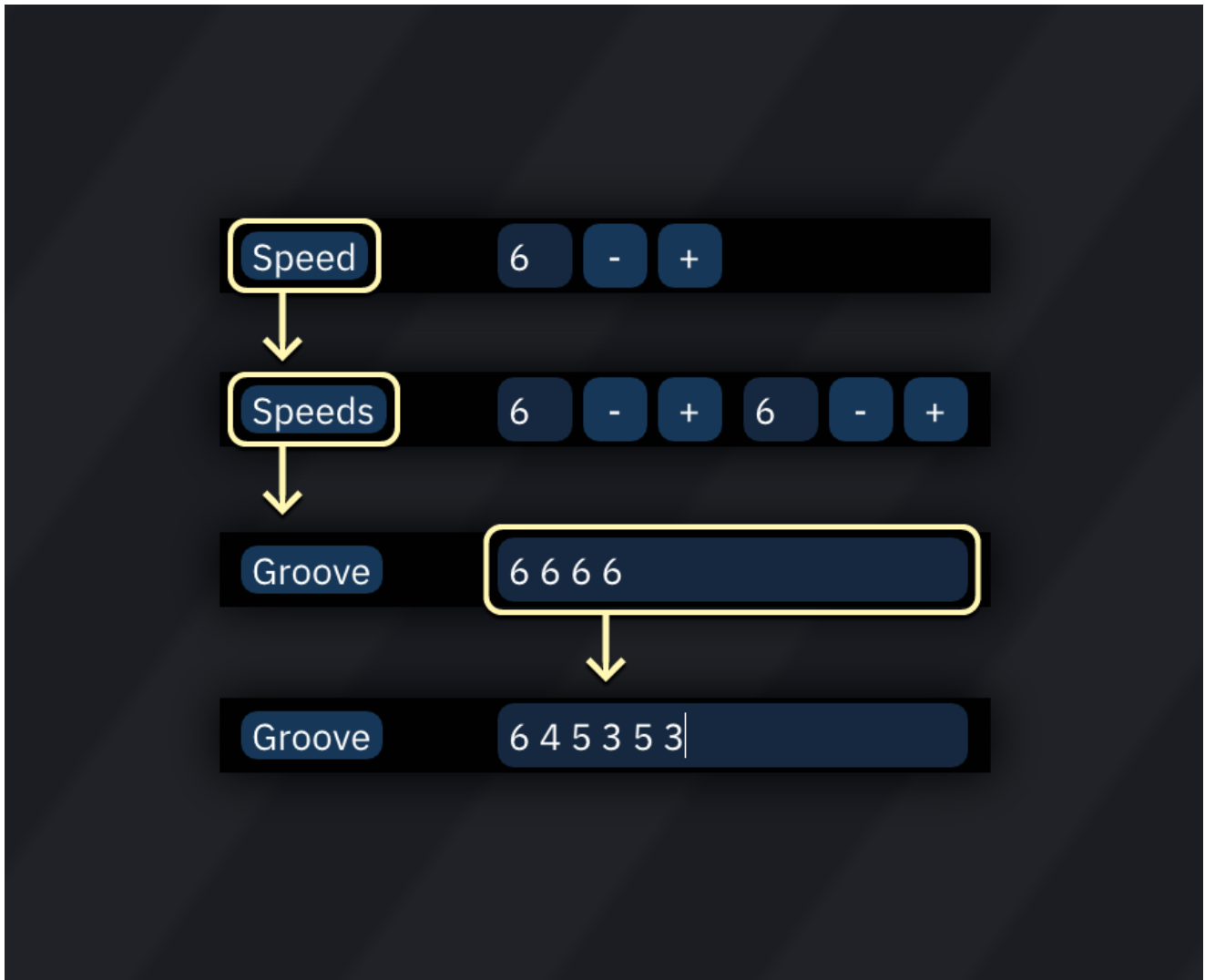
- **set:** changes matched data to this value.
- **add:** adds this value to matched data. it may be negative for subtraction. notes are calculated in semitones.
- **add (overflow):** as "add" above, but values will wrap around; for example, adding 13 to FF will result in 0C.
- **scale:** multiply value to this percentage; for example, scaling 1A by 150 results in 27. not available for "note".
- **clear:** erases matched data.

the following options also are available:

- **Add effect:** adds another Effect and Value to be replaced according to how they were found.
- **Remove effect:** removes last Effect and Value.
- **Effect replace mode:**
- **Replace matches only:** replaces only the effect columns that match.
- **Replace matches, then free spaces:** replaces matched effects; if there are effect columns without data, those will be filled in with the additional effect replacements.
- **Clear effects:** overwrites effect data with replacement effects.
- **Insert in free spaces:** replaces nothing; replacement effects are inserted in free effects columns when available.
- **Replace:** performs the query specified in the Find tab and replaces it as directed.

# grooves

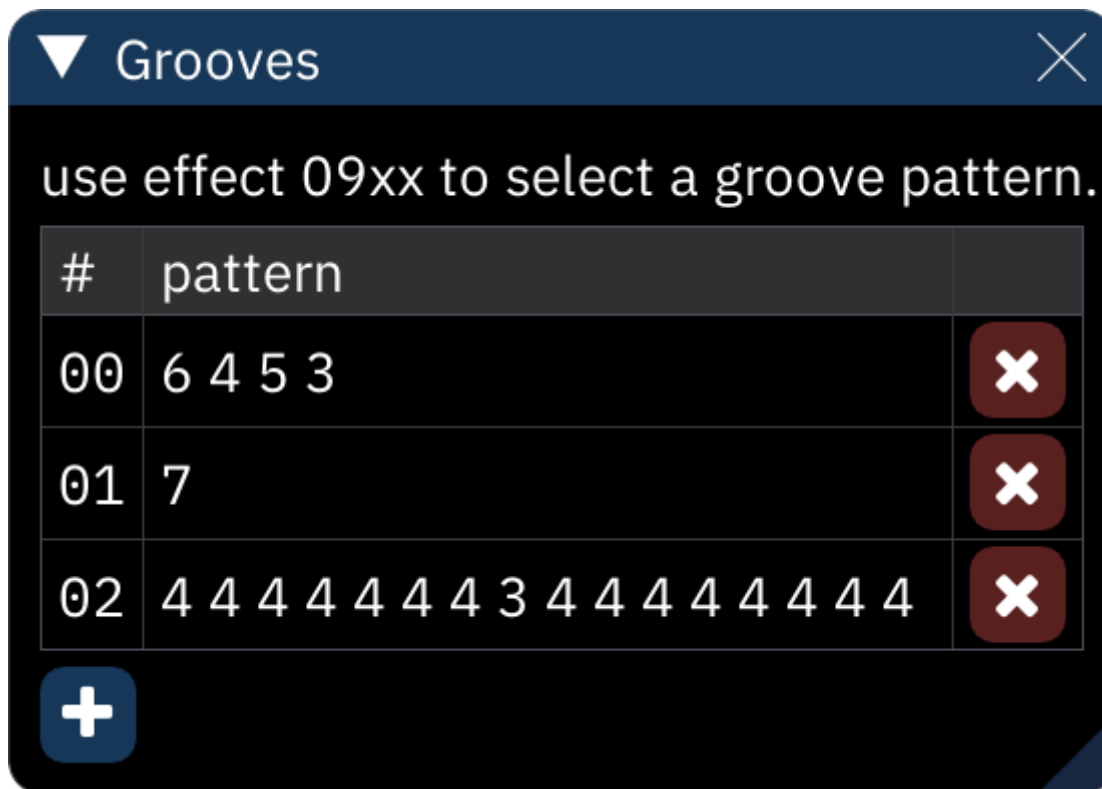
a **groove** is the equivalent of repeating 0Fxx effects on each row to get a cycle of speeds. for example, a groove of "6 4 5 3" makes the first row 6 ticks long, the next row 4 ticks, then 5, 3, 6, 4, 5, 3...



to set the song's groove:

- open the "Speed" window.
- click the "Speed" button so it becomes "Speeds" (effectively a groove of two speeds).
- click again so it becomes "Groove".
- enter a sequence of up to 16 speeds.





the "Grooves" window displays the list of groove patterns in the song.

- the + button adds a new groove pattern; click in the groove pattern to edit it.
- the × buttons remove them.

a single 09xx command will switch to the matching numbered groove pattern.

## tempo

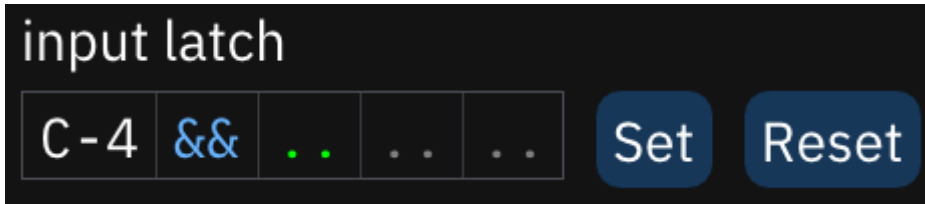
this is a non-exhaustive list of grooves and their equivalent tempo in BPM when using NTSC or PAL tick rates (60 or 50) and row highlight values of 4 and 16. for more accurate tempos, differing tick rates, or differing row highlight values, use this [groove calculator](https://pineight.com/ft/calcgroove.html) (https://pineight.com/ft/calcgroove.html) (courtesy of Damian Yerrick).

BPM NTSC	BPM PAL	GROOVE		BPM NTSC	BPM PAL	GROOVE
100.00	83.33	9		168.75	140.63	6, 5, 5
102.86	85.71	9, 9, 9, 8		171.43	142.86	6, 5, 5, 5
103.85	86.54	9, 9, 8		175.61	146.34	6, 5, 5, 5, 5, 5, 5, 5
105.88	88.24	9, 8		180.00	150.00	5
108.00	90.00	9, 8, 8		184.62	153.85	5, 5, 5, 5, 5, 5, 5, 4
109.09	90.91	9, 8, 8, 8		189.47	157.89	5, 5, 5, 4
112.50	93.75	8		192.86	160.71	5, 5, 4
114.29	95.24	8, 8, 8, 8, 8, 8, 8, 7		194.59	162.16	6, 4, 5, 4, 5, 4, 5, 4
116.13	96.77	8, 8, 8, 7		200.00	166.67	5, 4
118.03	98.36	9, 7, 8, 7, 8, 7, 8, 7		205.71	171.43	5, 4, 5, 4, 5, 4, 4, 4

BPM NTSC	BPM PAL	GROOVE	BPM NTSC	BPM PAL	GROOVE
120.00	100.00	8, 7	207.69	173.08	5, 4, 4
122.03	101.69	8, 7, 8, 7, 8, 7, 7, 7	211.76	176.47	5, 4, 4, 4
124.14	103.45	8, 7, 7, 7	218.18	181.82	5, 4, 4, 4, 4, 4, 4, 4
126.32	105.26	8, 7, 7, 7, 7, 7, 7, 7	225.00	187.50	4
128.57	107.14	7	232.26	193.55	4, 4, 4, 4, 4, 4, 4, 3
130.91	109.09	7, 7, 7, 7, 7, 7, 7, 6	240.00	200.00	4, 4, 4, 3
133.33	111.11	7, 7, 7, 6	245.45	204.55	4, 4, 3
135.00	112.50	7, 7, 6	248.28	206.90	5, 3, 4, 3, 4, 3, 4, 3
135.85	113.21	8, 6, 7, 6, 7, 6, 7, 6	257.14	214.29	4, 3
138.46	115.38	7, 6	266.67	222.22	4, 3, 4, 3, 4, 3, 3, 3
141.18	117.65	7, 6, 7, 6, 7, 6, 6, 6	270.00	225.00	4, 3, 3
142.11	118.42	7, 6, 6	276.92	230.77	4, 3, 3, 3
144.00	120.00	7, 6, 6, 6	288.00	240.00	4, 3, 3, 3, 3, 3, 3, 3
146.94	122.45	7, 6, 6, 6, 6, 6, 6, 6	300.00	250.00	3
150.00	125.00	6	327.27	272.73	3, 3, 3, 2
153.19	127.66	6, 6, 6, 6, 6, 6, 6, 5	337.50	281.25	3, 3, 2
156.52	130.43	6, 6, 6, 5	360.00	300.00	3, 2
158.82	132.35	6, 6, 5	385.71	321.43	3, 2, 2
160.00	133.33	7, 5, 6, 5, 6, 5, 6, 5	400.00	333.33	3, 2, 2, 2
163.64	136.36	6, 5	450.00	375.00	2
167.44	139.53	6, 5, 6, 5, 6, 5, 5, 5	900.00	750.00	1

# input latch

---

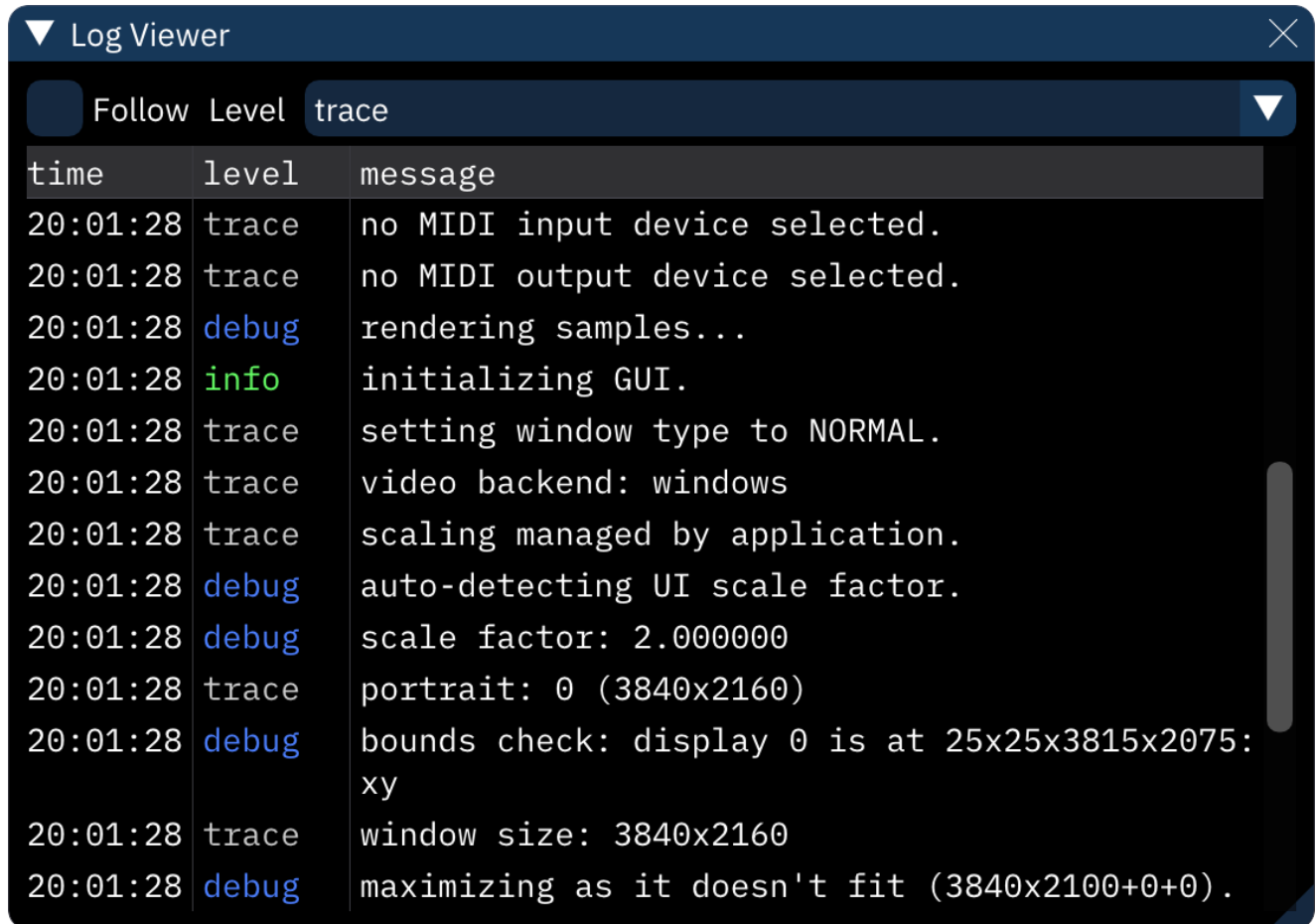


input latch determines which data are placed along with a note. as in the pattern view, the columns are note (not changeable), instrument, volume, effect type, and effect value.

- && fills in the currently selected instrument.
- . . ignores the column.
- all columns (except note) can be reset with a right-click.
- **Set**: sets latch according to the data found at the cursor.
- **Reset**: resets all columns to default (selected instrument, ignore others).
- only the first effect type and effect value may be latched.

# log viewer

the log viewer provides a look at Furnace's internal messages. this can be useful for chasing down problems.



if the **Follow** checkbox is enabled, the log will snap to the bottom and continually scroll to show the newest messages. if disabled, it will stay put on what's currently shown.

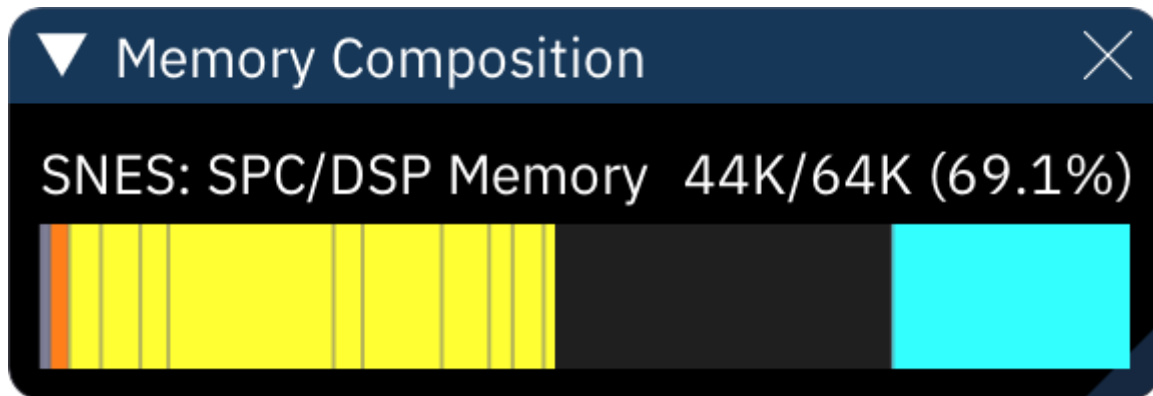
the **Level** dropdown determines the minimum importance of the messages displayed.

LEVEL	MESSAGE SHOWN
ERROR	serious problem
warning	may or may not be a problem
info	significant information
debug	general info about what Furnace is doing
trace	detailed info useful only to developers

# memory composition

---

this window displays the memory usage of chips that support memory (e.g. for samples).



# mixer

the "Mixer" dialog provides options for overall sound mixing.

## "Mixer" tab

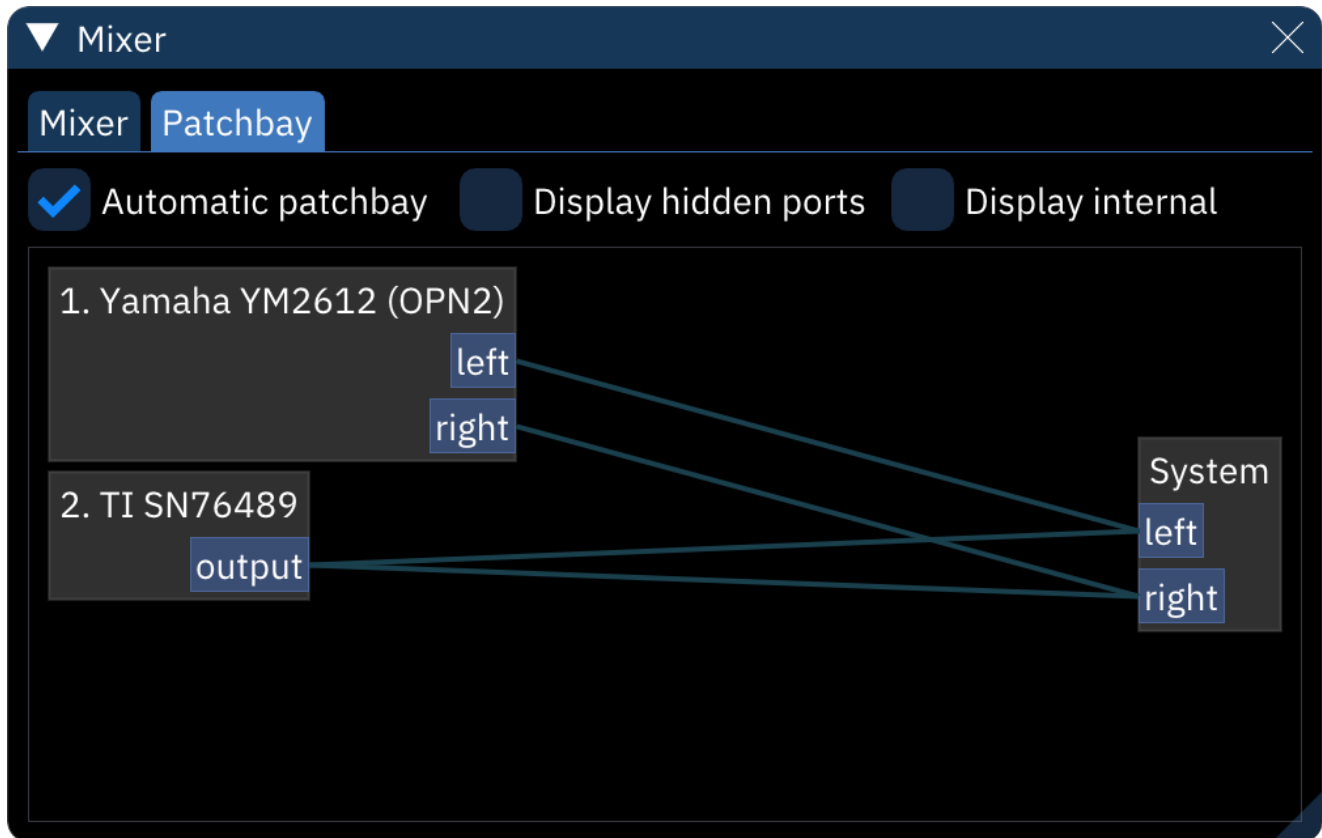


"Master Volume" controls the overall mix.

each chip has several options:

- **Invert:** flips the output wave.
- **Volume:** controls the chip's volume relative to other chips.
- **Panning:** left-right sound control.
- **Front/Rear:** front-rear sound control. only useful for setups with four or more speakers.

## "Patchbay" tab



- **Automatic patchbay:** make appropriate connections when adding, removing, or changing chips and chip settings.
- **Display hidden ports:** shows all available connection ports. the "System" unit actually has 16 ports; 1 maps to the left channel, and 2 maps to the right.
- **Display internal:** shows two additional units, one for sample previews and one for the metro nome sound.

the graph shows each existing unit along with their outputs, inputs, and the "patch cables" connecting them. connections can be made by dragging between an output and an input. right-clicking on a unit gives the option to disconnect all patches from that unit.

# operation mask

C-4	01	7F	04	72	delete
C-4	01	7F	04	72	pull delete
C-4	01	7F	04	72	insert
C-4	01	7F	04	72	paste
C-4	--	--	--	--	transpose (note)
---	01	7F	--	72	transpose (value)
C-4	01	7F	04	72	interpolate
C-4	01	7F	04	72	fade
C-4	01	7F	04	72	invert values
C-4	01	7F	04	72	scale
C-4	01	7F	04	72	randomize
C-4	01	7F	04	72	flip
C-4	01	7F	04	72	collapse/expand

the operation mask toggles which columns will be affected by the listed operations. as in the pattern view, the columns are note, instrument, volume, effect types, and effect values. the effect toggles apply to all effect columns.

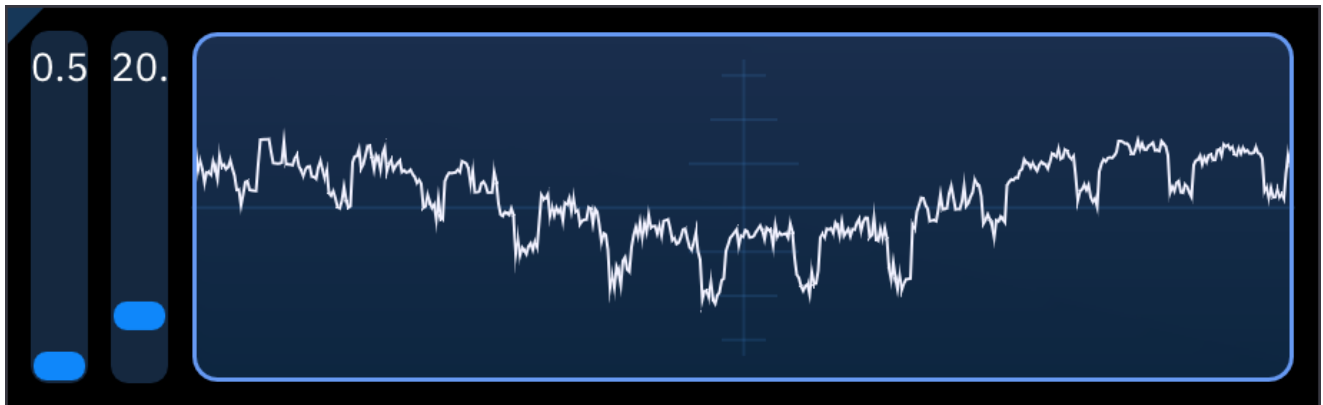
click any area to toggle it. a --- or -- means the operation will ignore any data in that column.



# oscilloscope

---

the Oscilloscope shows the audio output as a waveform.

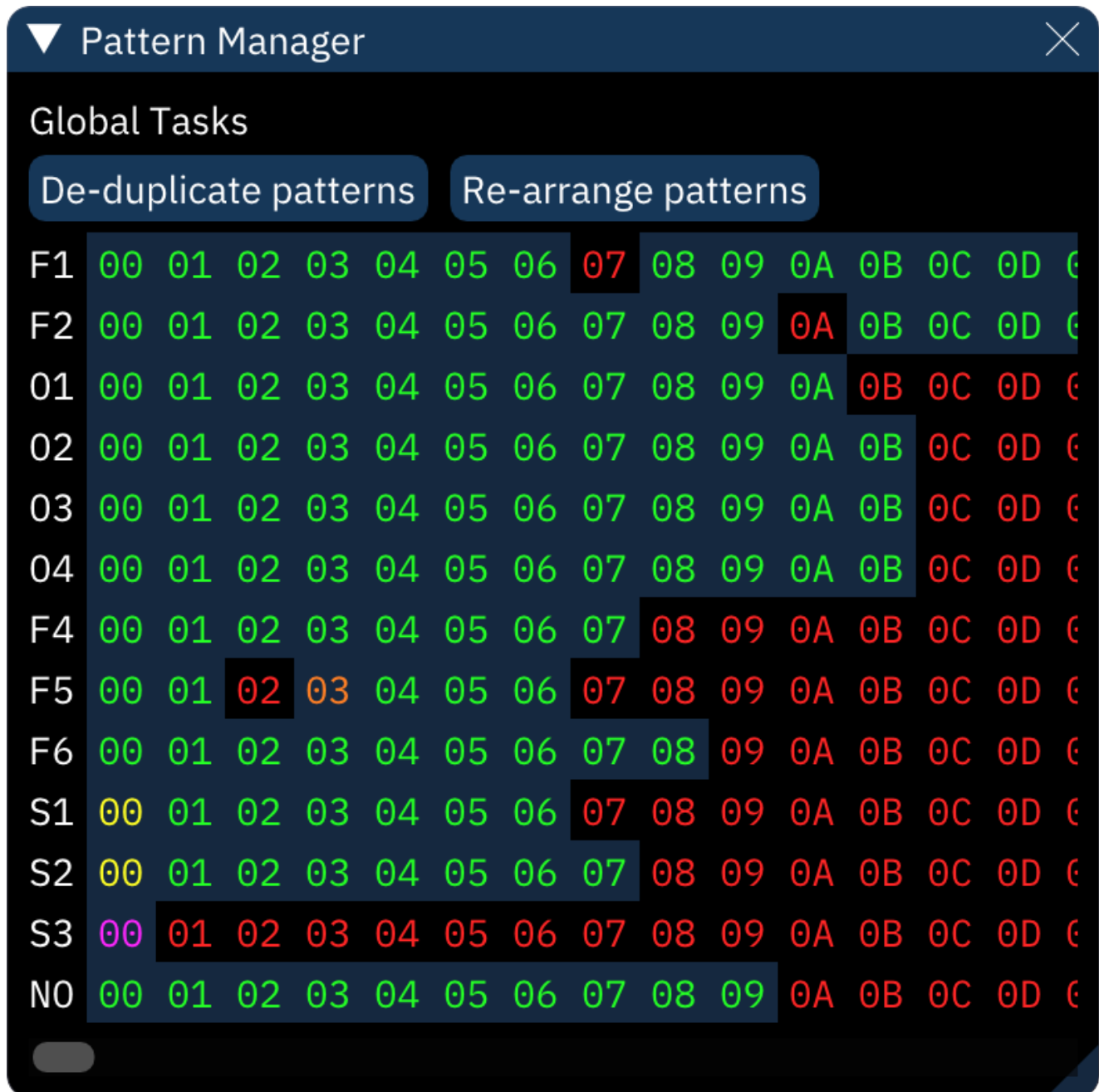


right-clicking on the oscilloscope toggles adjustment sliders:

- waveform height (zoom)
- window size (how much of the output to display) in milliseconds.

# pattern manager

the pattern manager is useful for cleaning up stray patterns and as an overview of pattern usage.



**De-duplicate patterns** looks for matching patterns, eliminates all but the first instance, and changes all references in the order list to match.

**Re-arrange patterns** rennumbers patterns to be in sequence, along with changing all references in the order list to match.

**Sort orders** rennumbers orders and re-arranges patterns so that they are in order.

**Make patterns unique** will clone all patterns that have been used more than once.

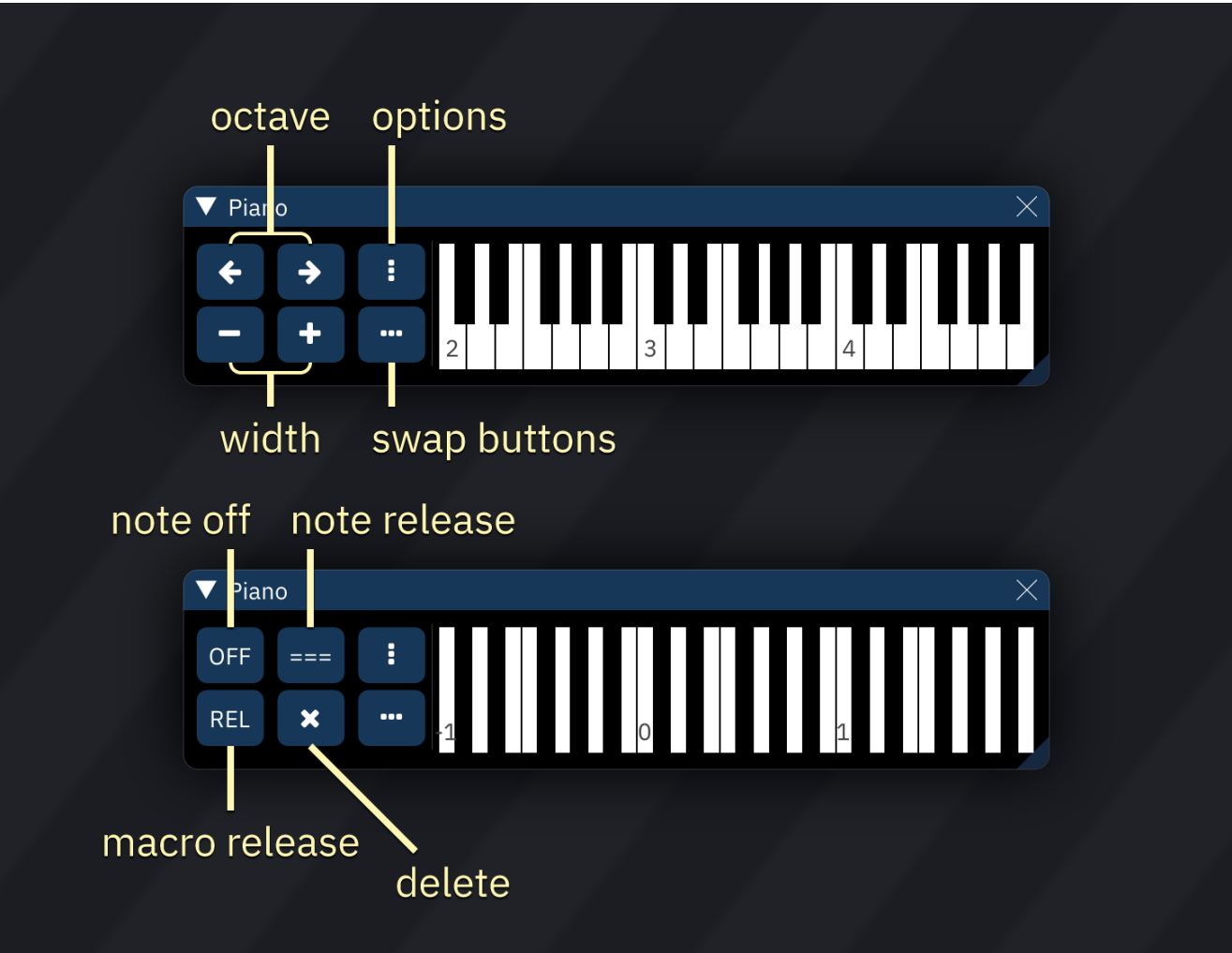
the pattern grid shows each channel and all its patterns. these are color-coded to show how much they're used in the song; these colors can be changed in Settings.

DEFAULT COLOR	NAME IN SETTINGS	MEANING
grey	Unallocated	pattern doesn't exist yet
red	Unused	exists but isn't in order list
green	Used	used only once in order list
yellow	Overused	used multiple times
orange	Really overused	used in half or more orders
magenta	Combo Breaker	the only used pattern in this channel!

right-clicking a pattern will permanently delete it.

# piano / input pad

the piano serves as a non-keyboard interface to input notes.



the buttons at the left do the following:

move one octave down	move one octave up	open options
fewer visible octaves	more visible octaves	swap buttons

when swapped, the buttons do the following:

input note off	input note release	open options
input macro release	delete	swap buttons

every C key is labelled with its octave.

right-clicking on the piano keys will make the buttons disappear; right-clicking again brings them back.

# options

---

key layout:

- **Automatic**
- **Standard**: black keys are 2/3 length.
- **Continuous**: black keys are full length.

**Share play/edit offset/range**: if disabled, the piano will keep different octave and range values for playback and non-playback states.

**\*\*Read-only** (can't input notes): prevents note entry.

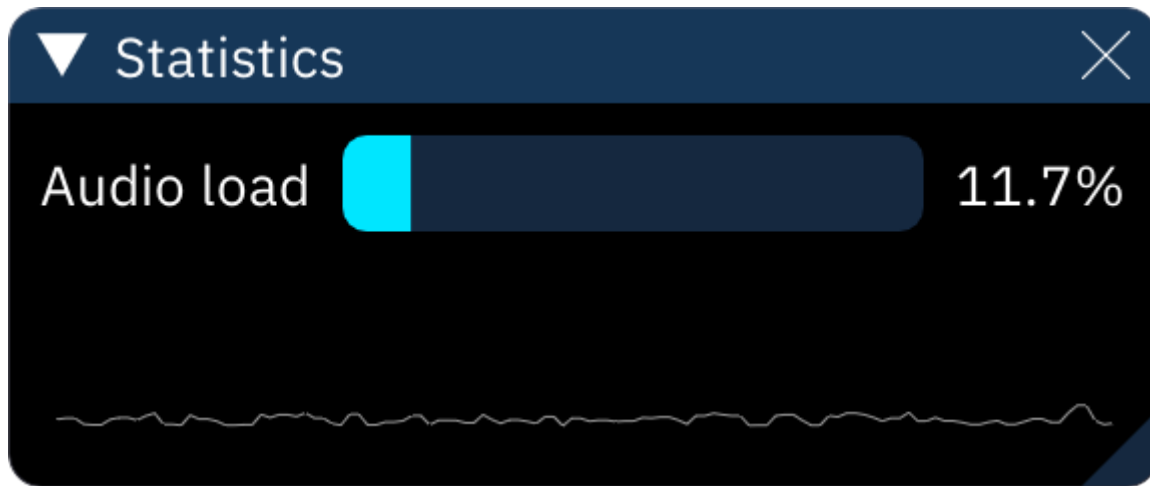
# register view

during playback, "Register View" shows the hex data involved with each chip's operation.

▼ Register View																	×
1. Game Boy																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	bf	00	39	07	00	ff	00	72	06	80	00	20	58	83	00	
20	3f	08	04	80	77	ff	8f	00	00	00	00	00	00	00	00	00	
30	ef	fe	b7	53	14	69	bc	ef	ff	b8	65	68	9a	bb	96	5c	

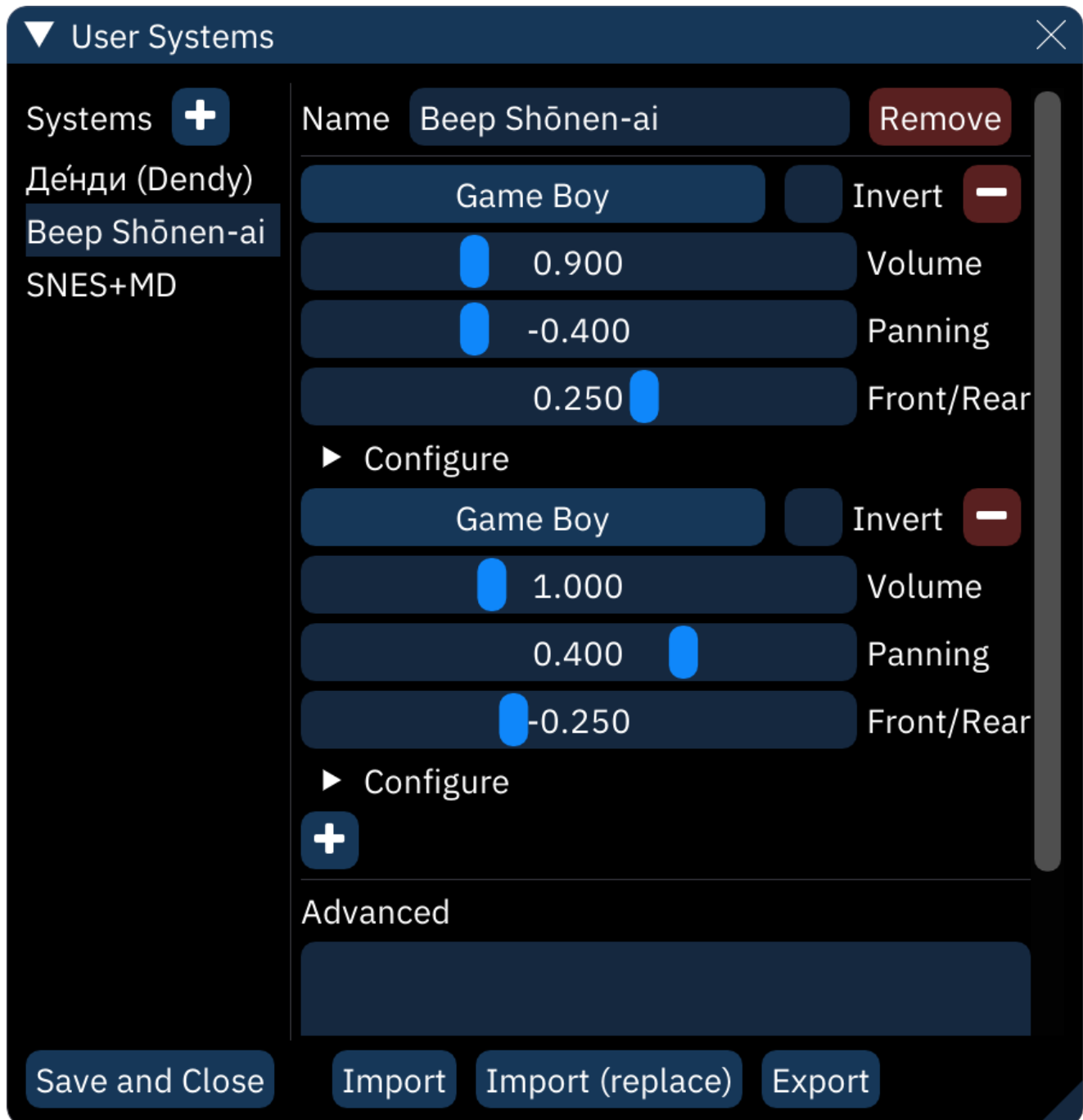
# statistics

the Statistics window shows current audio load (CPU used by emulation/playback) and a chart of audio load over the last two seconds.



# user systems

combinations of chips and chip configurations can be stored as **user systems** – presets that are easily accessed when starting a new song.



the + button at the top of the **Systems** list will add a new system.

next to the **Name** field, the **Remove** button removes the current system from the list.

chip configuration is exactly as in the [chip manager](#)<sub>360</sub> window.



the **Advanced** field stores additional settings that are set when a new song is started. these are listed in "option=value" format, one per line.

- tickRate: sets tick rate.
- chanMask: sets which channels to hide. written as a comma-separated list of integers

**Save and Close:** as it says.

**Import:** opens a dialog to select a .cfgu file, then adds its systems to the list.

**Import (replace):** opens a similar dialog, then clears the existing systems list and replaces it with the imported one.

**Export:** stores the current list of systems in a selected .cfgu file.

# oscilloscope (X-Y)

also called "vectorscope", this is similar to the normal oscilloscope in that it draws audio output as a waveform, but instead of being one-dimensional and going from left to right, it is plotted in two dimensions (usually X represents left output and Y represents right output).



right-clicking the X-Y oscilloscope window displays settings:

- **X Channel:** the output channel which will affect the X (horizontal) axis. default is 1 (left).
- **Invert:** flips the axis, therefore flipping the view horizontally.
- **Y Channel:** the output channel which will affect the Y (vertical) axis. default is 2 (right).
- **Invert:** flips the axis, therefore flipping the view vertically.
- **Zoom:** changes amplitude, making the plot bigger or smaller.
- **Samples:** the maximum number of samples to use for plotting.
- this setting may be replaced by another or removed in a future version.
- **Decay Time (ms):** sets how long it takes for the points/lines to fade away.
- **Intensity:** changes the brightness of the dots/lines.
- **Line Thickness:** sets how thick lines are.

click **OK** to return to the oscilloscope view.

# guides

---

this is a collection of user-contributed Furnace guides which may be useful during composition.

- [tuning samples](#)<sup>398</sup>
- [using samples with limited playback rates](#)<sup>396</sup>
- [choosing emulation cores](#)<sup>392</sup>
- [using OPLL patch macro](#)<sup>397</sup>
- [using AY/SAA hardware envelope](#)<sup>394</sup>

## other resources

---

- [FM Synthesis of Real Instruments](http://www.javelinart.com/FM_Synthesis_of_Real_Instruments.pdf) ([http://www.javelinart.com/FM\\_Synthesis\\_of\\_Real\\_Instruments.pdf](http://www.javelinart.com/FM_Synthesis_of_Real_Instruments.pdf)) : an in-depth tutorial on creating FM patches from scratch.

# choosing emulation cores

---

Furnace achieves the authentic sound of videogame hardware by emulating sound chips as accurately as possible, using **emulator cores**. in some cases there are multiple cores to choose from, each with different strengths and weaknesses. here are the major differences between them all.

- **YM2151 core:**
  - **ymfm**: default playback core. much less CPU usage than Nuked-OPM, but less accurate. recommended for users with mobile, last-gen or earlier hardware.
  - **Nuked-OPM**: default render core. much more accurate than ymfm, due to the emulator being based on an image of the die map taken from a real YM2151. very CPU heavy, only recommended for users with recent hardware.
- **YM2612 core:**
  - **Nuked-OPN2**: default core. lighter on the CPU than Nuked-OPM, can be used to simulate any variant of YM2612.
  - **ymfm**: same as ymfm above.
  - **YMF276-LLE**: a new core written by the author of the Nuked cores, specifically focused on YMF276 emulation. it is very slow and not useful for real-time playback. Produces audio comparable to output of Sega Mega Drive model 2 and later Fujitsu FM Towns computers, doesn't emulate DAC distortion of the original YM2612.
- **SN76489 core:**
  - **MAME**: default core. less accurate than Nuked, but with lower CPU usage. comes from the MAME emulator project.
  - **Nuked-PSG Mod**: more accurate, but not by that much. this originally started as an emulator for the YM7101 PSG sound generator, but was modified to emulate the SN7 as the MAME core was deemed unsatisfactory by some.
- **NES core:**
  - **puNES**: default core. it comes from a dedicated NES emulator.
  - **NSFplay**: higher CPU usage than puNES.
- **FDS core:**
  - **puNES**: default playback core. lower CPU usage and far less accurate.
  - **NSFplay**: default render core. higher CPU usage and much more accurate.
- **SID core:**
  - **reSID**: default playback core. a high quality emulation core. somewhat CPU heavy.
  - **reSIDfp**: default render core. improved version of reSID. the most accurate choice. very CPU heavy.
  - **dSID**: a lightweight open-source core used in DefleMask. not so accurate but it's very CPU light.
- **POKEY core:**
  - **Atari800 (mzpokeysnd)**: does not emulate two-tone mode.
  - **ASAP (C++ port)**: default core. the sound core used in the ASAP player. most accurate option.
- **OPN/OPNA/OPNB cores:**
  - **ymfm only**: lower CPU usage, less accurate FM.
  - **Nuked-OPN2 (FM) + ymfm (SSG/ADPCM)**: default cores. more accurate FM at the cost of more CPU load.
  - **YM2608-LLE**: a new core written by the author of the Nuked cores. high accuracy, but extremely high CPU usage, far beyond any other emulation core. (over 500% CPU time on 10th gen Intel Core i5!)
- **OPL/OPL2/Y8950 core:**

- **Nuked-OPL3**: high quality OPL emulation core. slightly off due to tiny differences between OPL and OPL3, but otherwise it is good.
- **ymfm**: this core is supposed to use less CPU than Nuked-OPL3, but for some reason it actually doesn't.
- **YM3812-LLE**: a new core written by the author of the Nuked cores. it features *extremely* accurate emulation.
  - this core uses a *lot* of CPU time. may not be suitable for playback!
- **OPL3 core**:
- **Nuked-OPL3**: high quality OPL emulation core.
- **ymfm**: this core is supposed to use less CPU than Nuked-OPL3, but for some reason it actually doesn't.
- **YMF262-LLE**: a new core written by the author of the Nuked cores. it features *extremely* accurate emulation.
  - this core uses even more CPU than YM3812-LLE. not suitable for playback or even rendering if you're impatient!
- **OPL4 core**:
- **Nuked-OPL3 (FM) + openMSX (PCM)**: high quality OPL4 emulation core.
- **ymfm**: this core is supposed to use less CPU than Nuked-OPL3, but for some reason it actually doesn't.
- **ESFM core**:
- **ESFMu**: the ESFM emulator. best choice but CPU intensive.
- **ESFMu (fast)**: this is a modification of ESFMu to reduce CPU usage at the cost of less accuracy.
- **OPLL core**:
- **Nuked-OPLL**: this core is accurate and the default.
- **emu2413**: a less accurate core that uses less CPU.
- **AY-3-8910/SSG core**:
- **MAME**: default core.
- **AtomicSSG**: SSG core extracted from YM2608-LLE.
- **WonderSwan core**:
- **asiekierka new core**: default core. highest accuracy and efficiency.
- **Mednafen**: slower and less accurate. included for compatibility with older modules.

# AY-3-8910 / AY8930 / SAA1099 envelope guide

---

The AY-3-8910 programmable sound generator, aside from normal 4-bit volume control, has an hardware volume envelope. This feature that allows for defining the shape of the volume envelope at arbitrary speed according to 8 preset envelope shapes. One may think, what is any upside of hardware envelope? Well, it's somewhat independent of tone/noise generators, and since it goes so high in frequency, it can be used melodically! This guide explains how to make best use of the AY/SAA envelope.

## AY-3-8910 / AY8930

---

In the instrument editor:

- Add a single tick to the "Waveform" macro with only envelope turned on. This will disable any output, but don't worry.
- Add a single tick to the "Envelope" macro and select enable.

If you play a note now, you will hear a very high-pitched squeak. This is because you must set envelope period, which is the frequency at which the hardware envelope runs. You can do it in two ways:

- 23xx and 24xx effects (envelope coarse and fine period);
- 29xx auto-envelope period effect and macros.

Auto-envelope works via numerator and denominator. In general, the higher the numerator, the higher the envelope pitch. The higher the denominator, the lower the envelope pitch. Why are there both of these? Because the envelope generator might be used to mask the tone output (i.e. affect the square wave as well). To do it, set the "Waveform" macro values to both tone and envelope. The higher the denominator value, then the lower the envelope pitch relative to the square wave output, and similarly with the numerator. With the square-and-envelope setting, a lot of wild, de tuned synth instruments can be made.

Back to the hardware envelope itself. Depending on the "Envelope" macro value, different envelope shapes can be obtained. The most basic one, 8, is a sawtooth wave. The `direction` value will invert the envelope, producing the reverse sawtooth. The `alternate` value produces an interesting pseudo-triangular wave, similar to halved sine. That one can also be reversed. `Hold` option disables the envelope.

*Warning:* The envelope pitch resolution is fairly low; at high pitches it will be detuned. Because of this, it's used mostly for bass.

*Warning:* There is only one hardware envelope generator. You can't use two pitches or two wave forms at once.

# SAA1099

---

SAA envelope works a bit differently. It doesn't have its own pitch; instead, it relies on the channel 2/5 pitch. It also has many more parameters than the AY envelope. To use it:

- Go to waveform macro and add a single tick set to 0 (unless you want to have a square wave mask).
- Set up an envelope macro. Turn on enabled, loop, and depending on the desired shape, cut and direction. Resolution will give you higher pitch range than on the AY.
- Place two notes in the pattern editor. One in channel 2 will control the envelope pitch. The other in channel 3 can be any note you wish; it's just to enable the envelope output.

## examples

---

- [Demoscene-type Beat by Duccinator](https://www.youtube.com/watch?v=qcBgmpPrLUA) (<https://www.youtube.com/watch?v=qcBgmpPrLUA>)
- [Philips SAA1099 Test by Duccinator](https://www.youtube.com/watch?v=IBh2gr09zjs) (<https://www.youtube.com/watch?v=IBh2gr09zjs>)
- [Touhou Kaikidan: Mystic Square title theme by ZUN](https://www.youtube.com/watch?v=tUKei7Pz0Fw) (<https://www.youtube.com/watch?v=tUKei7Pz0Fw>) : Rare instance of AY envelope used for drums, it can be used to mask the noise generator output too

# using samples with limited playback rates

---

some sample-based chips have a limited number of available sample playback rates. when working with these chips, notes entered in the pattern editor will play back at the closest available rate... which might be perfect, or might be several semitones off. the solution is to prepare samples to work around this.

for example: using the NES, a C-4 note in the PCM channel means the associated sample will play back at a rate of 8363Hz. let's say we want to use a slap bass sample recorded at a rate of 22050Hz in which the audible pitch is A-2. let's also say that when we put a C-4 note in the tracker, we want to hear the bass play at what sounds like C-3, transposed three semitones higher than the recorded pitch.

here's how to make this example work:

- load up the sample and open it in the sample editor.
- the Note selector will show "F-6"; add the three semitones mentioned above to make it "G#6". the Hz will change to 26217.
- use the Resample button. in the pop-up dialog, type in 8363, then click Resample.
- select the instrument from the instrument list, and in the pattern editor, enter a C-4 in the PCM channel. it should sound like a slap bass playing a C-3 note.

the NES PCM frequency table shows the sixteen notes can be played. if a D-4 is entered, the slap bass will be heard at D-3 as desired. what if we want to hear a C#3, though?

- load up the original sample in a new slot and open it in the sample editor.
- the Note selector will show "F-6"; this time we add four semitones to make it "A-6". the Hz will change to 27776.
- just like before, use the Resample button. in the little pop-up, type in 8363 (yes, the NES's C-4 rate), then click Resample.
- select the instrument from the instrument list and open it in the instrument editor. turn on "Use sample map".
- in the leftmost column, find C#5. click in the next column and enter the number of the second sample. in the next column to the right, click the "C#4" and hit the key for C-4 to change it.
- in the pattern editor, enter a C#4; it should sound like a C#3!
- try adding another entry in the sample map so the note D#4 plays the second sample at D-4.



# using OPLL patch macro

---

YM2413's biggest flaw (or, rather, cost-saving feature) was that it could use only one user-defined instrument at once. It wasn't monotimbral; you could use 15 built-in presets and 5 built-in drum tones freely, but for these going off the beaten path, it surely was limiting. However, there is one technique, as amazing as simple: **mid-note preset switching**.

the idea is to use the first patch to put the envelope in an unintended state for the second patch so that it sounds different, with a higher or lower modulation level. the sustain level defines at which "envelope level" the envelope will switch to the sustain state (or release depending on envelope type). if the first patch is used to put the envelope into sustain at a higher or lower envelope state than intended for the second patch, it'll still be in sustain/release but at a higher or lower level than it should be at that point.

therefore, much more variety can be forced out, without using custom instruments. As of July 2023, Furnace is the only tool supporting this feature. It is accessed in 'Macros' tab in OPLL instrument editor.

For example, try putting the first macro value as 14 (acoustic bass preset), followed by 4 (flute preset). This way you will get distortion guitar-like sound this is nothing like other 2413 preset! There are many combination to test out, which is highly recommended (I can only say, 12->1 or 12->4 produces sound similar to the well-known 4-op FM mallet brass)

## drums using this technique

---

Using OPLL's drum mode, described in systems/opll.md, you gain access to 5 hardcoded drum tones at the expense of 3 melodic FM channels. Patch switching eliminates that, as using it, it's also possible to construct percussive sounds, some even fuller than their drum mode counter parts!

In short, noise portion of drums (as in hi-hats), can be created of the very high pitched pseudo-distortion guitar, described as above. For kicks, snares, toms and claps, more effort is needed, however using volume and arpeggio macros will help.

## examples

---

- [Lman-Clubster cover by Mahbod](https://www.youtube.com/watch?v=jfHs7tSyjXI) (<https://www.youtube.com/watch?v=jfHs7tSyjXI>)
- [OPLL Nation by Mahbod](https://www.youtube.com/watch?v=ou6pEfxByeE) (<https://www.youtube.com/watch?v=ou6pEfxByeE>)

# tuning samples

---

loading a new sample into Furnace is easy... but getting it transposed and tuned to match the song can be tricky at first.

it's important to remember that the "Hz" and "Note" as shown in the sample editor are unrelated to the note heard in the sample itself. a sample shown as having a "Note" of C-4 will use a sample rate of 4181, even though it may contain a note played at a different pitch than C.

for this example, we'll use a sample of a note played at E and recorded at 22050Hz.

- if needed, use the "Create instrument from sample" button to make an instrument to use in the track.
- calculate the semitone difference in Hz between the note your recorded sample is playing and C. in this example, the nearest C is 4 semitones down from E.
- set "Note" to 4 semitones lower than it shows. in this case, it starts at F-6, so set it to C#6.
  - or, use a pitch calculator like <https://www.omnicalculator.com/other/semitone> (<https://www.omnicalculator.com/other/semitone>) . input Frequency 1 as 22050Hz, input -4 semitones, and receive a Frequency 2 of 17501.10Hz. enter that value into "Hz".
- now, using the "Preview sample" button should play the note at C. entering an E in the pattern will now play it at or near E.
- if the sample still sounds out of tune, adjust "Hz" or "Fine" to bring it in line.

if notes seem "capped" – for example, playing anything over D-6 sounds like a D-6 – those notes exceed the maximum sample playback rate for the chip. the only solution is to use "Resample" to change the sample to a lower rate.