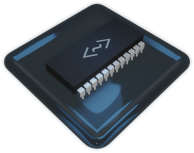


Furnace
User Manual

tildearrow and contributors

Furnace user manual

this documentation is a work in progress! expect several sections to be incomplete.

attribution

writers:

- tildearrow
- freq-mod
- nicco1690
- DeMOSic
- cam900
- host12prog
- WindowxDeveloper

other:

- **3-pattern/keyboard.png** licensed under CC-BY-SA 3.0.
- this is a derivative of **KB United States.svg** by Denelson83 under the same license.

introduction

Furnace is a tool which allows you to create music using emulated sound chips from the 8/16-bit era.
For a full list of soundchips that Furnace supports, please see **this list (page 44)** .

It has a music tracker interface. think of a piano roll, or a table that scrolls up and plays the notes.

Another core feature of Furnace is its windowing system, similar to that of GEMS or Deflemask, but with a few more features.

Sound generation

Furnace generates sound from 3 different main types of sound sources.
- Instruments are the most standard and most used type of sound source in Furnace.
The instrument format is how you can specify parameters and macros for certain channels on certain soundchips, as well as binding samples and wavetables to a format that you can sequence on the note grid.
See **4-instrument (page 19)** for more details.
- Wavetables are the way that you create custom waveform shapes for the HuC6280, Seta X1-010, WonderSwan, any PCM chip with wavetable synthesizer support, etc.
Wavetables only work in the sequencer if you bind them to an instrument.
See **4-instrument (page 19)** and **5-wave (page 41)** for more details.
- Samples are how you play back raw audio streams (samples) on certain channels, on certain soundchips, and in some cases, in certain modes.
To sequence a sample, you do not need to assign it to an instrument, however, to resample samples (change the speed of a sample), you need to bind it to a Sample instrument.
See **6-sample (page 42)** and **4-instrument (page 19)** for more details.

Interface/other

Furnace is built to have a user-friendly interface that is intentionally made so that it is quick and easy to get around when working in Furnace.
However, we understand that the interface may not be the easiest to learn, depending on how you learn, so there is documentation on it as well.

See **2-interface (page 4)** and **3-pattern (page 10)** to view said documentation.

interface

the Furnace user interface is where the job gets done.

basic mode

Furnace comes with a "basic mode" that can be toggled through the "settings" menu. it disables certain features in Furnace that may look intimidating or confusing for newcomers. if you find that a certain feature of furnace is missing, see if this setting is enabled or not.

among the features that cannot be accessed in this mode are:
* file menu:
* chip manipulation options (add, configure, change, remove chips)
* edit menu:
* paste special...
* operation masking
* input latch
* find and replace
* speed window:
* virtual tempo
* divider
* song length
* song info window:
* manual system naming
* tuning options
* right-clicking on the pattern window:
* gradient/fade...
* scale
* randomize
* invert values
* flip selection
* collapse
* expand
* these windows:
* mixer
* grooves
* channels
* pattern manager
* chip manager
* compatibility flags

UI components

the user interface consists of several components. this paper describes some of them.

windows

TODO: image

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

to move a window, press and hold the mouse button while on title bar or any empty space on it.
then drag your mouse, and release it to stop moving.

to resize a window, drag any of the bottom corners (marked by triangular tabs).

to collapse a window, click on the triangle in the title bar.
clicking again expands it.

to close a window, click on the X at the top right corner.

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 five options will appear, allowing you to select where and how to dock that window.
the options are:

```
        UP
LEFT CENTER RIGHT
      DOWN
```

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

if you drag the window to CENTER, the window will be maximized to cover the workspace (if you do this on the workspace), or it will appear as another tab (if you do this on a window).

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.

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 single option: "Hide tab bar".
selecting this option will hide the tab bar of that window.
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.

text fields

TODO: image

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-Z`: undo
- `Ctrl-Y`: redo
- `Ctrl-A`: select all

(replace Ctrl with Command on macOS)

number input fields

TODO: image

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

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

sliders

TODO: image

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

alternatively, right-clicking or Ctrl-clicking or a slider (Command-click on macOS) will turn it into a number input field for a short period of time, allowing you to input fine values.

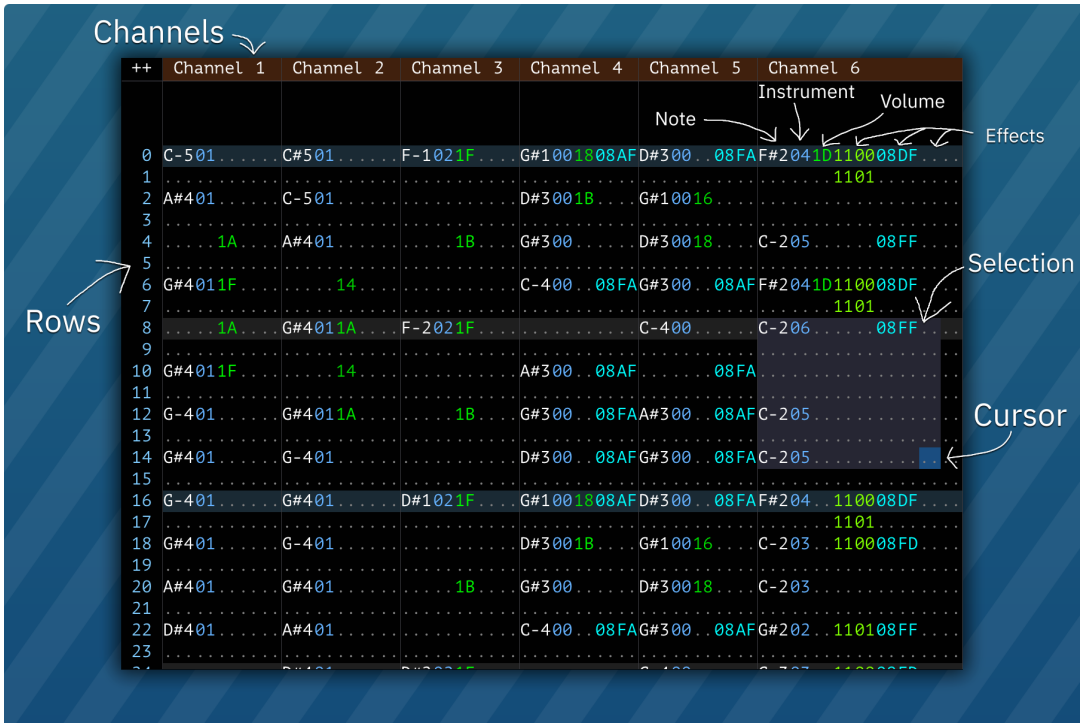
global keyboard shortcuts

this is a list of available shortcuts that may be used at any time, regardless of currently focused window.

KEY	ACTION
Ctrl-O	open file
Ctrl-S	save file
Enter	play/stop song
F5/F6	"
Shift-Enter	play from cursor position
F7	"
Ctrl-Enter	play one row
NumPad /	decrease octave
NumPad *	increase octave

patterns

the pattern view allows you to edit the song.



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

1. note
2. instrument
3. volume
4. effect and effect value (several)

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

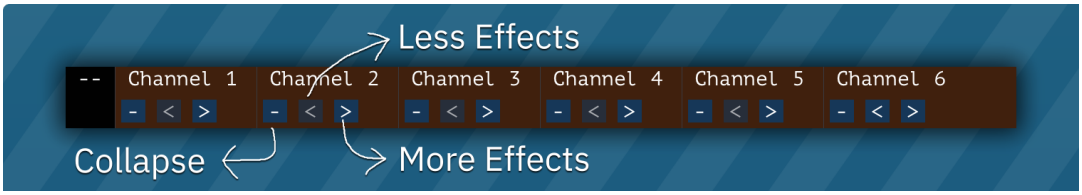
managing channels

you may mute channels, toggle solo mode, collapse channels or even hide them.

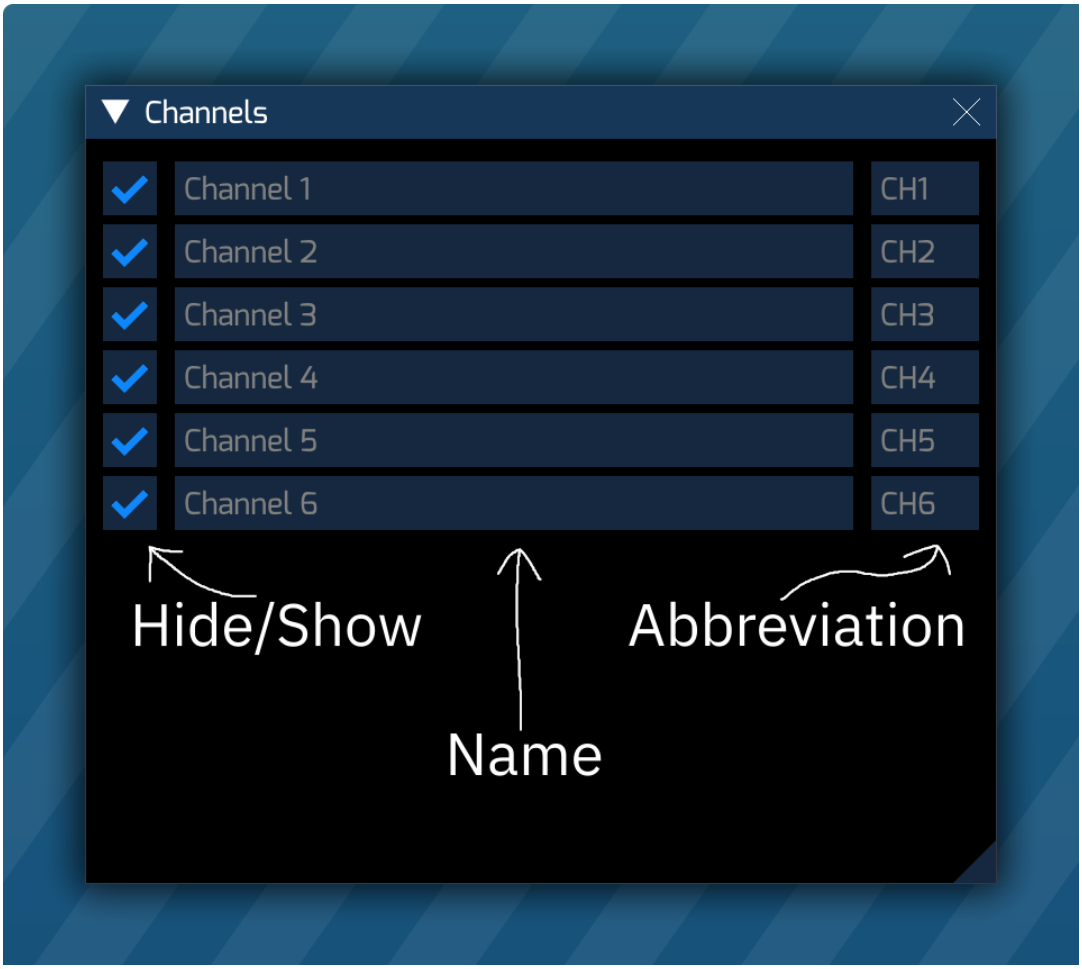
clicking on a channel name mutes that channel.

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

clicking the ++ at the top left corner of the pattern view displays additional buttons for channel configuration:



to rename and/or hide channels, see the Channels window (window > channels).



cursor and selection

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

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

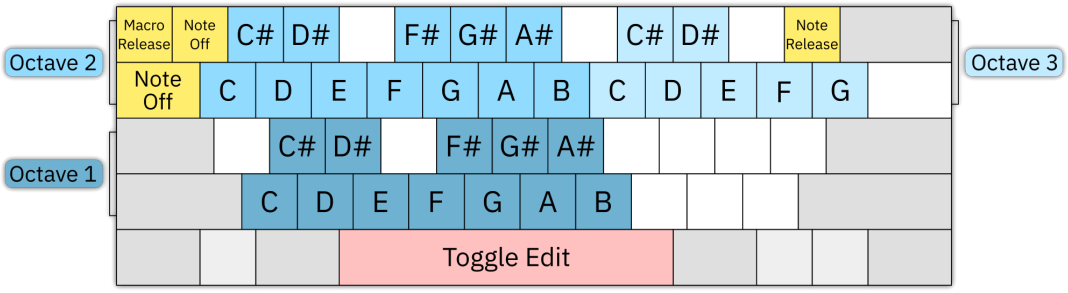
keyboard layout

shortcuts

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
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)

KEY	ACTION
Ctrl-F4	transpose selection (+1 octave)
Space	toggle note input (edit)

note input



- pressing any of the respective keys will insert a note at the cursor's location, and then advance it by the Edit Step.
- note off turns off the last played note in that channel (key off on FM; note cut otherwise).
- note release triggers macro release (and in FM channels it also triggers key off).
- macro release does the same as above, but does not trigger key off in FM channels.

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.

for a list of effects **click here** **(page 14)** .

effect list

most of the effect numbers are from ProTracker/FastTracker 2.
however, effects are continuous, which means you only need to type it once and then stop it with an effect value of `00`.

- `00`xy: arpeggio. after using this effect the channel will rapidly switch between `note`, `note+x` and `note+y`.
- `01`xx: slide up.
- `02`xx: slide down.
- `03`xx: note portamento.
- a note must be present for this effect to work.
- `04`xy: vibrato. x is the speed, while y is the depth.
- maximum vibrato depth is ±1 semitone.
- `07`xy: tremolo. x is the speed, while y is the depth.
- maximum tremolo depth is -60 volume steps.
- `08`xy: set panning. x is the left channel and y is the right one.
- not all chips support this effect.
- `80`xx: set panning (linear). this effect behaves more like other trackers:
- `00` is left.
- `80` is center.
- FF is right.
- not all chips support this effect.
- 81xx: set volume of left channel (from `00` to FF).
- not all chips support this effect.
- 82xx: set volume of right channel (from `00` to FF).
- not all chips support this effect.
- `09`xx: set speed 1.
- `0A`xy: volume slide.
- if x is 0 then this is a slide down.
- if y is 0 then this is a slide up.
- `0B`xx: jump to pattern.
- `0C`xx: retrigger note every xx ticks.
- this effect is not continuous.
- `0D`xx: jump to next pattern.

- `0F`xx: set speed 2.

- 9xxx: set sample position to xxx*0x100.

- not all chips support this effect.

- Cxxx: change song Hz.

- xxx may be from `000` to `3ff`.

- E0xx: set arpeggio tick.

- this sets the number of ticks between arpeggio values.
- 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.
- E3xx: set vibrato direction. xx may be one of the following:
- `00`: up and down.
- `01`: up only.
- `02`: down only.
- E4xx: set vibrato range in 1/16th of a semitone.
- E5xx: set pitch. `80` is 0 cents.
- range is ±1 semitone.
- EAxx: toggle legato.
- EBxx: set sample bank.
- does not apply on Amiga.
- ECxx: note off after xx ticks.
- EDxx: delay note by xx ticks.
- EExx: send external command.
- this effect is currently incomplete.
- F0xx: change song Hz by BPM value.
- F1xx: single tick slide up.
- F2xx: single tick slide down.
- F3xx: fine volume slide up (64x slower than `0`Axy).
- F4xx: fine volume slide down (64x slower than `0`Axy).
- F5xx: disable macro.
- see macro table at the end of this document for possible values.
- F6xx: enable macro.
- F8xx: single tick volume slide up.
- F9xx: single tick volume slide down.
- FAxy: fast volume slide (4x faster than `0`Axy).
- if x is 0 then this is a slide down.
- if y is 0 then this is a slide up.
- FFxx: end of song/stop playback.

additionally each chip has its own effects. **click here for more details (page 44)** .

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
10	extra 5
11	extra 6
12	extra 7
13	extra 8
---	----------
20	**operator 1 macros** - AM
21	AR
22	DR
23	MULT
24	RR
25	SL
26	TL

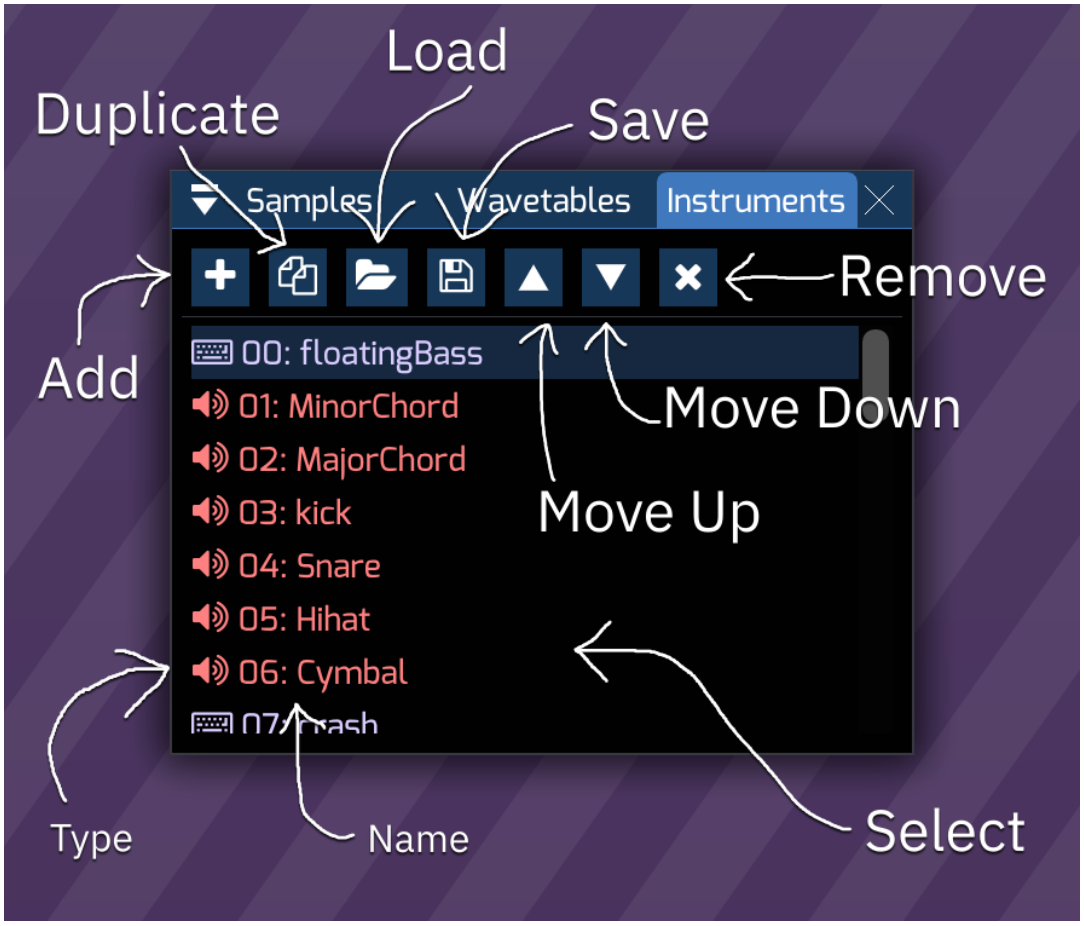
ID	MACRO
27	DT2
28	RS
29	DT
2A	D2R
2B	SSG-EG
2C	DAM
2D	DVB
2E	EGT
2F	KSL
30	SUS
31	VIB
32	WS
33	KSR
---	----------
40	operator 2 macros
60	operator 2 macros
80	operator 2 macros

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

EX	FM	OPM	OPZ	OPLL	AY-3-8910	AY8930	LYNX	C
D	NoiseF	Noise-Freq			Noise-Freq	Noise-Freq	Duty/Int	D
W		LFO Shape	LFO Shape	Patch	Wave-form	Wave-form		W f
1		AMD	AMD			Duty		P N
2		PMD	PMD		Envel-ope	Envel-ope		P c a
3	LFOSpd	LFO Speed	LFO Speed		AutoEn-vNum			S c

EX	FM	OPM	OPZ	OPLL	AY-3-8910	AY8930	LYNX	C
						Au-toEn-vNum		
A	ALG	ALG	ALG		AutoEn-vDen	Au-toEn-vDen		
B	FB	FB	FB			Noise AND		
C	FMS	FMS	FMS			Noise OR		
D	AMS	AMS	AMS					
4	Op-Mask	Op-Mask						T C
5			AMD2					
6			PMD2					
7			LFO2Speed					
8			LFO2Shape					

instrument list



click on an instrument to select it.

double-click to open the instrument editor.

instrument editor

every instrument can be renamed and have its type changed.

depending on the instrument type, there are currently 13 different types of an instrument editor:

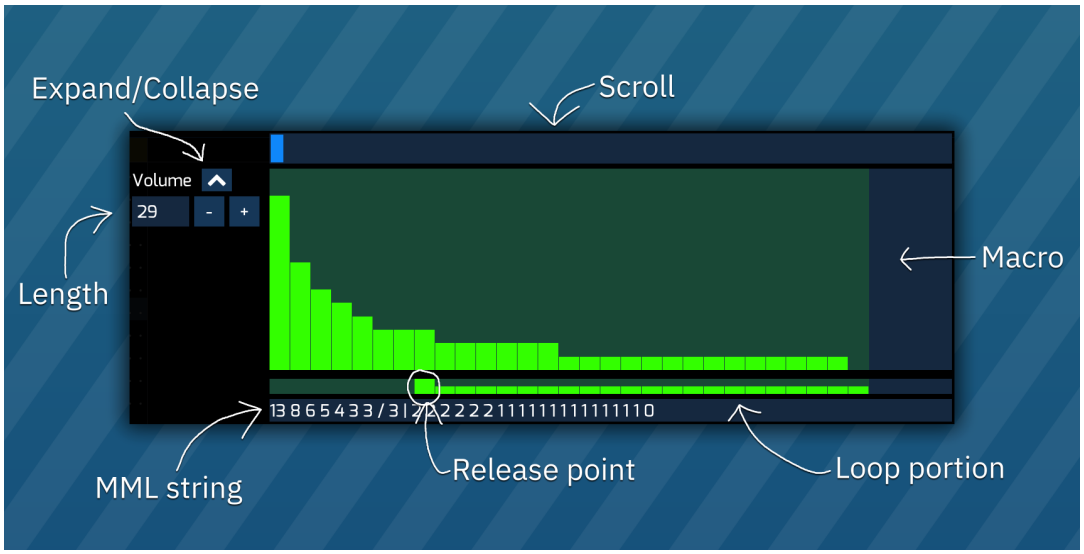
- **FM synthesis (page 27)** - for use with YM2612, YM2151 and FM block portion of YM2610.
- **Standard (page 35)** - for use with NES and Sega Master System's PSG sound source and its derivatives.
- **Game Boy (page 29)** - for use with Game Boy APU.
- **PC Engine/TurboGrafx-16 (page 32)** - for use with PC Engine's wavetable synthesizer.

- **WonderSwan (page 39)** - for use with WonderSwan's wavetable synthesizer.
- **AY8930 (page 22)** - for use with Microchip AY8930 E-PSG sound source.
- **Commodore 64 (page 25)** - for use with Commodore 64 SID.
- **SAA1099 (page 33)** - for use with Philips SAA1099 PSG sound source.
- **TIA (page 36)** - for use with Atari 2600 chip.
- **AY-3-8910 (page 24)** - for use with AY-3-8910 PSG sound source and SSG portion in YM2610.
- **Amiga/sample (page 23)** for controlling Amiga and other sample based synthsizers like YM2612's Channel 6 PCM mode, NES channel 5, Sega PCM, X1-010 and PC Engine's sample playback mode.
- **Atari Lynx (page 30)** - for use with Atari Lynx handheld console.
- **VERA (page 37)** - for use with Commander X16 VERA.
- **Seta/Allumer X1-010 (page 40)** - for use with Wavetable portion in Seta/Allumer X1-010.
- **Konami SCC/Bubble System WSG (page 34)** - for use with Konami SCC and Wavetable portion in Bubble System's sound hardware.
- **Namco 163 (page 31)** - for use with Namco 163.
- **Konami VRC6 (page 38)** - for use with VRC6's PSG sound source.

macros

one common feature to instruments is macros (also known as sequences).

these run on every tick and are useful for controlling parameters automatically.



to change the loop portion/point, click on the bar under the macro.
right click on it to disable macro loop.

to change the release point, shift-click the bar under the macro.
shift-right click on it to remove the release point.

AY8930 instrument editor

AY8930 instrument editor consists of 10 macros.

- [Volume] - volume levels sequence
- [Arpeggio]- pitch sequence
- [Noise frequency] - AY8930 noise generator frequency sequence
- [Waveform] - selector of sound type - pulse wave tone, noise or envelope generator
- [Duty cycle] - duty cycle of a pulse wave sequence
- [Envelope] - allows shaping an envelope
- [Auto envelope numerator] - sets the envelope to the channel's frequency multiplied by numerator
- [Auto envelope denominator] - sets the envelope to the channel's frequency multiplied by denominator
- [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

Amiga/PCM sound sourceinstrument editor

PCM instrument editor consists of four macros and sample selector:

Amiga/sample

- [Initial sample] - specifies which sample should be assigned to the instrument, or the first one in the sequence

Macros

- [Volume] - volume sequence WARNING: it works only on Amiga system, as of version 0.5.5!!
- [Arpeggio] - pitch sequence
- [Waveform] - sample sequence

AY-3-8910 instrument editor

AY-3-8910 instrument editor consists of 7 macros.

- [Volume] - volume levels sequence
- [Arpeggio]- pitch sequence
- [Noise frequency] - AY-3-8910 noise generator frequency sequence
- [Waveform] - selector of sound type - square wave tone, noise or envelope generator
- [Envelope] - allows shaping an envelope
- [Auto envelope numerator] - sets the envelope to the channel's frequency multiplied by numerator
- [Auto envelope denominator] - ets the envelope to the channel's frequency multiplied by denominator

C64 SID instrument editor

C64 instrument editor consists of two tabs: one controlling various parameters of sound channels and macro tab containing seven macros:

C64

- [Waveform] - allows selecting a waveform. NOTE: more than one waveform can be selected at once, logical AND mix of waves will be produced, with an exception of a noise waveform, it can't be mixed.
- [Attack] - determines the rising time for the sound. The bigger the value, the slower the attack. (0-15 range)
- [Decay]- Determines the diminishing time for the sound. The higher the value, the longer the decay. It's the initial amplitude decay rate. (0-15 range)
- [Sustain] - Determines the diminishing time for the sound. The higher the value, the longer the decay. This is the long "tail" of the sound that continues as long as the key is depressed. (0-15 range)
- [Release] - Determines the rate at which the sound disappears after KEY-OFF. The higher the value, the longer the release. (0-15 range)
- [Ring Modulation] - enables the ring modulation affecting the instrument.
- [Duty] - specifies the width of a pulse wave. (0-4095 range)
- [Oscillator Sync] - enables the oscillator hard sync. As one oscillator finishes a cycle, it resets the period of another 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 synced oscillator's frequency.
- [Enable filter] - enables analogue filter affecting the instrument
- [Initialize filter] - initializes the filter with the specified parameters:
- [Cutoff] - defines the "intensity" of a filter, to put in in layman terms (0-2047 range)
- [Resonance] - defines an additional controlled amplification of that cutoff frequency, creating a secondary peak forms and colors the original pitch. (0-15 range)
- [Filter mode] - determined the filter mode NOTE: SID's filter is mulitimode, you can mix different modes together (like low and high-pass filters at once) CH3-OFF disables the channel 3, for no reason whatsoever lmao
- [Volume Macrio is a Cutoff macro] - turns a volume macro in a macros tab into a filter cutoff macro.

- [Absolute Cutoff macro] - changes the behaviour of a cutoff macro from the old-style, compatible to much more define-able.
- [Absolute Duty macro] - changes the behaviour of a duty cycle macro from the old-style, compatible to much more define-able.

Macros

- [Volume/Cutoff] - volume sequence (WARNING: Volume sequence is global for ALL three channels!!)
- [Arpeggio] - pitch sequence
- [Duty cycle] - pulse duty cycle sequence
- [Waveform] - select the waveform used by instrument
- [Filter mode] - select the filter mode/squence
- [Resonance] - filter resonance sequence
- [Special] - ring and oscillator sync selector

FM synthesis instrument editor

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 paramets of operator 1
- [Macros (OP2)] - for macros controlling FM paramets of operator 2
- [Macros (OP3)] - for macros controlling FM paramets of operator 3
- [Macros (OP4)] - for macros controlling FM paramets of operator 4
- [Macros] - for miscellanous macros controlling volume, argeggio and YM2151 noise generator.

FM

FM synthesizers Furnace supports are for-operator, meaning it takes four oscillators to produce a single sound. Each operator is controlled by a dozen of sliders:

- [Attack Rate (AR)] - determines the rising time for the sound. The bigger the value, the faster the attack. (0-31 range)
- [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. (0-31 range)
- [Secondary Decay Rate (DR2)/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. (0-31 range)
- [Release Rate (RR)] - Determines the rate at which the sound disappears after KEY-OFF. The higher the value, the shorter the release. (0-15 range)
- [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. (0-15 range)
- [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 (KSR)] - A parameter that determines the degree to which the envelope execution speed increases according to the pitch. (0-3 range)
- [Frequency Multiplier (MULT)] - Determines the operator frequncy in relation to the pitch. (0-15 range)
- [Fine Detune (DT)] - Shifts the pitch a little (0-7 range)

- [Coarse Detune (DT2)] - Shifts the pitch by tens of cents (0-3 range) WARNING: this parameter affects only YM2151 sound source!!!
- [Hardware Envelope Generator (SSG-EG)] - Executes the built-in envelope, inherited from AY-3-8910 PSG. Speed of execution is controlled via Decay Rate. WARNING: this parameter affects only YM2610/YM2612 sound source!!!
- [Algorithm (AL)] - Determines how operators are connected to each other. (0-7 range)
- [Feedback (FB)] - Determines the amount of signal whick operator 1 returns to itself. (0-7 range)
- [Amplitude Modulation (AM)] - Makes the operator affected by LFO.
- [LFO Frequency Sensitivity] - Determines the amount of LFO frequency changes. (0-7 range)
- [LFO Amplitude Sensitivity (AM)] - Determines the amount of LFO frequency changes. (0-3 range)

Macros

Macros define the squence of values passed to the given parameter. Via macro, aside previously mentioned parameters, the following can be controlled:

- LFO Frequency
- LFO waveform selection WARNING: this parameter affects only YM2151 sound source!!!
- Amplitude Modulation Depth WARNING: this parameter affects only YM2151 sound source!!!
- Frequency Modulation Depth WARNING: this parameter affects only YM2151 sound source!!!
- Arpeggio Macro: pitch change sequence in semitones. Two modes are available:
  Absolute (defult) - Executes the pitch with absolute change based on the pitch of the actual note.
  Fixed - Executes at the pitch specified in the sequence regardless of the note pitch.
- Noise Frequency: specifies the noise frequency in noise mode of YM2151's Channel 8 Operator 4 special mode.

Loop
You can loop the execution of part of a sequence. Left-click anywhere on the Loop line at the bottom of the editor to create a loop. You can move the start and end points of the loop by dragging both ends of the loop.

Game Boy instrument editor

GB instrument editor consists of two tabs: one controlling envelope of sound channels and macro tab containing only four macros:

Game Boy

- [Volume] - this slider affect the channel volume (range 0-15)
- [Envelope length] - this slider specifies the envelope decay/attack (range 0-7)
- [Sound length] - this slider cuts off the sound after specified length, overriding the previous slider's value
- [UP an DOWN radio buttons] - these buttons alter the behaviour of a second slider. Up makes it specify the envelope attack, down the decay. WARNING: for envelope attack to have any effect, volume should be at the lower rates!

Macros

- [Volume] - volume sequence
- [Arpeggio] - pitch sequence
- [Duty cycle] - pulse wave channels duty cycle sequence
- [Waveform] - ch3 wavetable sequence

Atari Lynx instrument editor

Atari Lynx instrument editor consists of only three macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequencer
- [Duty/Int] - bit pattern for LFSR taps and integration.

Audio generation description

Atari Lynx to generate sound uses 12-bit linear feedback shift register with configurable tap. Nine separate bits can be enable to be the source of feedback.
Namely bits 0, 1, 2, 3, 4, 5, 7, 10 and 11. To generate ANY sound at least one bit MUST be enable.

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.

Namco 163 instrument editor

Namco 163 instrument editor consists of two tabs: one controlling various parameters for waveform initialize and macro tab containing 10 macros.

Namco 163

- [Initial Waveform] - Determines the initial waveform for playing.
- [Initial Waveform position in RAM] - Determines the initial waveform position will be load to RAM.
- [Initial Waveform length in RAM] - Determines the initial waveform length will be load to RAM.
- [Load waveform before playback] - Determines the load initial waveform into RAM before playback.
- [Update waveforms into RAM when every waveform changes] - Determines the update every different waveform changes in playback.

Macros

- [Volume] - volume levels sequence
- [Arpeggio]- pitch sequence
- [Waveform pos.] - sets the waveform source address in RAM for playback (single nibble unit)
- [Waveform] - sets waveform source for playback immediately or update later
- [Waveform len.] - sets the waveform source length for playback (4 nibble unit)
- [Waveform update] - sets the waveform update trigger behavior for playback
- [Waveform to load] - sets waveform source for load to RAM immediately or later
- [Wave pos. to load] - sets address of waveform for load to RAM (single nibble unit)
- [Wave len. to load] - sets length of waveform for load to RAM (4 nibble unit)
- [Waveform load] - sets the waveform load trigger behavior

NEC PC Engine instrument editor

PCE instrument editor consists of only three macros, almost like TIA:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequence
- [Waveform] - spicifies wavetables sequence

It also has wavetable synthesizer support, but unfortunately, it clicks a lot when in use on the HuC6280.

Philips SAA1099 instrument editor

SAA1099 instrument editor consists of five macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequence
- [Duty cycle/ Noise] - noise generator frequency
- [Waveform] - selector between tone and noise
- [Envelope] - specifies the envelope generator shape

Konami SCC/Bubble System WSG instrument editor

SCC/Bubble System WSG instrument editor consists of only three macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequence
- [Waveform] - spicifies wavetables sequence

Standard instrument editor

SMS and NES instrument editor consists of only three macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequencr
- [Duty cycle] - spicifies duty cycle and noise mode for NES channels
  NOTE: it obviously has no effect on Sega Master System

Atari TIA instrument editor

TIA instrument editor consists of only three macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequencr
- [Waveform] - 1-bit polynomial pattern type sequence

VERA instrument editor

VERA instrument editor consists of only four macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequence
- [Duty cycle] - pulse duty cycle sequence
- [Waveform] - select the waveform used by instrument

VRC6 instrument editor

The VRC6 (regular) instrument editor consists of only three macros:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequence
- [Duty cycle] - specifies duty cycle for pulse wave channels

VRC6 (saw) instrument editor

This channel has its own instrument, a (currently, as of 0.6) one-of-a-kind thing in Furnace.

The only differences from this instrument type from compared to the regular instrument are that it:
- has a volume range of 0-63 instead of 0-15.
- has no duty cycle range

WonderSwan instrument editor

WS instrument editor consists of only four macros, similar to PCE but with different volume and noise range:

- [Volume] - volume sequence
- [Arpeggio] - pitch sequencr
- [Noise] - noise LFSR tap sequence
- [Waveform] - spicifies wavetables sequence

X1-010 instrument editor

X1-010 instrument editor consists of 7 macros.

- [Volume] - volume levels sequence
- [Arpeggio]- pitch sequence
- [Waveform] - spicifies wavetables sequence
- [Envelope Mode] - allows shaping an envelope
- [Envelope] - spicifies envelope shape sequence, it's also wavetable.
- [Auto envelope numerator] - sets the envelope to the channel's frequency multiplied by numerator
- [Auto envelope denominator] - sets the envelope to the channel's frequency divided by denominator

wavetable editor

Wavetable synthesizers, in context of Furnace, are sound sources that operate on extremely short n-bit PCM streams. By extremely short, no more than 256 bytes. This amount of space is nowhere near enough to store an actual sampled sound, it allows certain amount of freedom to define a waveform shape. As of Furnace 0.5.8, wavetable editor affects PC Engine, WonderSwan and channel 3 of Game Boy.

Furnace's wavetable editor is rather simple, you can draw the waveform using mouse or by pasting an MML bit stream in the input field. Maximum wave width (length) is 256 bytes, and maximum wave height (depth) is 256. NOTE: Game Boy, PCE, WonderSwan and Bubble System can handle max 32 byte waveforms, X1-010 can handle max 128 byte waveforms as of now, with 16-level height for GB, X1-010 Envelope, WS, Bubble System and N163, and 32-level height for PCE. If a larger wave is defined for these chips, it will be squashed to fit within the constraints of the chips.

wavetable synthesizer

Furnace contains a mode for wavetable instruments that 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 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 waveform is changed while the channel is playing. On certain frequencies, this can be avoided, but not on most, unfortunately.

samples

In the context of Furnace, a sound sample (usually just referred to as a sample) is a string of numbers that hold sampled PCM audio.

In Furnace, these samples can be generated by importing a .wav (think of it as an higher quality MP3) file.

supported chips

as of Furnace 0.6, 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/HuC6280
- Amiga/Paula
- SegaPCM
- Neo Geo/Neo Geo CD/YM2610 (ADPCM channels only)
- Seta/Allumer X1-010
- Atari Lynx
- MSM6258 and MSM6295
- YMU759/MA-2 (last channel only)
- QSound
- ZX Spectrum 48k (1-bit)
- RF5C68
- WonderSwan
- tildearrow Sound Unit
- VERA (last channel only)
- Y8950 (last channel only)
- Konami K007232
- a few more that I've forgotten to mention

compatible sample mode

effect 17xx enables/disables compatible sample mode whether 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.

use of this mode is discouraged in favor of Sample type instruments.

notes

due to limitations in some of those sound chips, some restrictions exist:

- Amiga: 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, and loop position isn't supported (only entire sample).
- 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.
- Neo Geo (ADPCM-A): no looping supported. your samples will play at ~18.5KHz.
- Neo Geo (ADPCM-B): no loop position supported (only entire sample), and the maximum frequency is ~55KHz.
- YM2608: the maximum frequency is ~55KHz.
- 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.

furthermore, many of these chips have a limited amount of sample memory. check memory usage in window > statistics.

the sample editor

You can actually tweak 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`.

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.

The changes you make will be applied as soon as you've committed them to your sample, but they can be undoed and redoed, just like text.

sound chips

this is a list of sound chips that Furnace supports, including effects.

- **Yamaha YM2203 (page 107)**
- **Yamaha YM2413 (page 75)**
- **Yamaha YM2608 (page 110)**
- **Neo Geo/YM2610 (page 113)**
- **Taito Arcade/YM2610B (page 116)**
- **Yamaha YM2612 (page 119)**
- **Yamaha YMZ280B (page 122)**
- **ZX Spectrum Beeper (page 123)**

Furnace also reads .dmf files with the **Yamaha YMU759 (page 121)** system, but...

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...

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.

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.

effects

- 20xx: set channel mode. xx may be one of the following:
- 00: square
- 01: noise
- 02: square and noise
- 03: envelope
- 04: envelope and square
- 05: envelope and noise
- 06: envelope and square and noise
- 07: 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 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.
- 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.

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 best known for being used in the Covox Sound Master, which didn't sell well either. It also observed very minimal success in Merit's CRT-250 machines, but only as a replacement for the AY-3-8910.

emulation of this chip in Furnace is now complete thanks to community efforts and hardware testing, which an MSX board called Darky has permitted.

effects

- 12xx: set channel duty cycle. xx is a value between 00 and 08.
- `00`: 3.125%
- `01`: 6.25%
- `02`: 12.5%
- `03`: 25%
- `04`: 50%
- `05`: 75%
- `06`: 87.5%
- `07`: 93.75%
- `08`: 96.875%
- `20`xx: set channel mode. xx may be one of the following:
- `00`: square
- `01`: noise
- `02`: square and noise
- `03`: envelope
- `04`: envelope and square
- `05`: envelope and noise
- `06`: envelope and square and noise
- `07`: 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.

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 32x16 wavetables.

effects

- `10xx`: change wave.

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).

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 to 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 to 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.
- 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)
- 3xxx: set duty cycle. xxx range is 000-FFF
- 4xxx: set cutoff. xxx range is 000-7FF.

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.

Famicom Disk System

the Famicom Disk System is an expansion device for the Famicom (known as NES outside Japan), a popular console from the '80's.
as it name implies, it allowed people to play games on specialized floppy disks that could be rewritten 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
- why is this mapping so unnatural? because that's how DefleMask does it (yeah, as you may have guessed this effect is mostly for compatibility reasons)...

Game Boy

the Nintendo Game Boy is one of the most successful portable game systems ever made.

with stereo sound, two pulse channels, a wave channel and a noise one it packed some serious punch.

effects

- 10xx: change wave.
- 11xx: set noise length. xx may be one of:
- 0: long
- 1: short
- 12xx: set duty cycle (from 0 to 3).
- 13xy: setup sweep (pulse channels 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.

Sega Genesis/Mega Drive

a video game console that showed itself as the first true rival to Nintendo's video game market near-monopoly in the US during the '80's.

this console is powered by two sound chips: the **Yamaha YM2612 (page 119)** and **a derivative of the SN76489 (page 89)** .

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 mutliplier.
- 17xx: enable PCM channel.
- this only works on channel 6.
- **this effect is there for compatibility reasons** - it is otherwise recommended to use Sample type instruments (which automatically enable PCM mode when used).
- 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.
- 20xy: set PSG noise mode.
- x controls whether to inherit frequency from PSG channel 3.
    - 0: use one of 3 preset frequencies (C: A-2; C#: A-3; D: A-4).
    - 1: use frequency of PSG channel 3.
- y controls whether to select noise or thin pulse.
    - 0: thin pulse.
    - 1: noise.

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 for now

Atari Lynx/MIKEY

the Atari Lynx is a 16 bit handheld console developed by (obviously) Atari Corporation, and initially released in September of 1989, with the world-wide release being in 1990.

while it was an incredible handheld for the time (and a lot more powerful than a Game Boy), it unfortunately meant nothing in the end due to the 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.

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 (×0, ×1, ×2, ×3, ×4, ×5, ×7, ×10, and ×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.
- hard 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 (0 to FFF).
- this is a bitmask.
- 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.

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 0-3 for the pulse channels.

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 (0 to 5).
- only in internal (capacitor-based) envelope mode.
- 13xx: set group decay (0 to 11).
- only in internal (capacitor-based) envelope mode.

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.

effects

...

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...)

effects

- 20xx: set chip output rate.

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 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.

Furnace supports loading waveforms into RAM and waveform playback simultaneously, and channel limit is dynamically changeable with effect commands.

you must load waveform to RAM first for playback, as its load behavior auto-updates when every waveform changes.

both waveform playback and load command work independently per each channel columns.
(Global) commands don't care about the channel columns for work commands and its load behavior is independent with per-channel column load commands.

effects

- 10xx: set waveform for playback.
- 11xx: set waveform position in RAM for playback (single nibble unit).
- 12xx: set waveform length in RAM for playback (04 to FC, 4 nibble unit).
- 130x: set playback waveform update behavior (0: off, bit 0: update now, bit 1: update when every waveform is changed).
- 14xx: set waveform for load to RAM.
- 15xx: set waveform position for load to RAM (single nibble unit).
- 16xx: set waveform length for load to RAM (04 to FC, 4 nibble unit).

- 170x: set waveform load behavior (0: off, bit 0: load now, bit 1: load when every waveform is changed).
- 180x: set channel limit (0 to 7, x + 1).
- 20xx: (Global) set waveform for load to RAM.
- 21xx: (Global) set waveform position for load to RAM (single nibble unit).
- 22xx: (Global) set waveform length for load to RAM (04 to FC, 4 nibble unit).
- 230x: (Global) set waveform load behavior (0: off, bit 0: load now, bit 1: load when every waveform is changed).

Namco WSG | Namco C15 | Namco C30

a family of wavetable synth sound chips used by Namco in their arcade machines (Pacman 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).

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 Famicom. it is a five-channel sound generator: 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.
- this may be used to attenuate the triangle and noise channels.
- will not work if a sample is playing.
- 12xx: set duty cycle or noise mode of channel.
- may be 0-3 for the pulse channels and 0-1 for the noise channel.
- 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; 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), availables pitches are limited and loop point is ignored.

length counter table

VAL	RAW	NTSC	PAL	DENDY	NTSC 5-STEP	PAL 5-STEP	DENDY 5-STEP
00	10	83ms	100ms	84ms	104ms	125ms	105ms
01	254	2.1s	2.5s	2.1s	2.6s	3.2s	2.7s
02	20	166ms	200ms	168ms	208ms	250ms	210ms
03	2	17ms	20ms	17ms	21ms	25ms	21ms
04	40	333ms	400ms	336ms	417ms	500ms	421ms
05	4	33ms	40ms	34ms	42ms	50ms	42ms
06	80	667ms	800ms	673ms	833ms	1.0s	841ms
07	6	50ms	60ms	50ms	63ms	75ms	63ms
08	160	1.3s	1.6s	1.3s	1.7s	2.0s	1.7s
09	8	67ms	80ms	67ms	83ms	100ms	84ms
0A	60	500ms	600ms	505ms	625ms	750ms	631ms
0B	10	83ms	100ms	84ms	104ms	125ms	105ms
0C	14	117ms	140ms	118ms	146ms	175ms	147ms
0D	12	100ms	120ms	101ms	125ms	150ms	126ms
0E	26	217ms	260ms	219ms	271ms	325ms	273ms
0F	14	117ms	140ms	118ms	145ms	175ms	147ms
10	12	100ms	120ms	101ms	125ms	150ms	126ms
11	16	133ms	160ms	135ms	167ms	200ms	168ms
12	24	200ms	240ms	202ms	250ms	300ms	252ms
13	18	150ms	180ms	151ms	188ms	225ms	189ms

VAL	RAW	NTSC	PAL	DENDY	NTSC 5-STEP	PAL 5-STEP	DENDY 5-STEP
14	48	400ms	480ms	404ms	500ms	600ms	505ms
15	20	167ms	200ms	168ms	208ms	250ms	210ms
16	96	800ms	960ms	807ms	1.0s	1.2s	1.0s
17	22	183ms	220ms	185ms	229ms	275ms	231ms
18	192	1.6s	1.9s	1.6s	2.0s	2.4s	2.0s
19	24	200ms	240ms	202ms	250ms	300ms	252ms
1A	72	600ms	720ms	606ms	750ms	900ms	757ms
1B	26	217ms	260ms	219ms	271ms	325ms	273ms
1C	16	133ms	160ms	135ms	167ms	200ms	168ms
1D	28	233ms	280ms	235ms	292ms	350ms	294ms
1E	32	267ms	320ms	269ms	333ms	400ms	336ms
1F	30	250ms	300ms	252ms	313ms	375ms	315ms

reference: **NESdev**

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, 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.

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 (sadly Furnace only supports stereo) and some other things.

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).
- 15xx: set operator 4 level.
- only in 4-op mode (OPL3).
- 16xy: set multiplier of operator.
- x is the operator (1-4; last 2 operators only in 4-op mode).

- y is the mutliplier.
- 17xx: set vibrato depth. the following values are accepted:
- 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 (1-4; last 2 operators only 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 (1-4; last 2 operators only 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 (1-4; last 2 operators only in 4-op mode). a value of 0 means "all operators".
- y is the value.
- 52xy set RR of operator.
- x is the operator (1-4; last 2 operators only in 4-op mode). a value of 0 means "all operators".
- y is the value.
- 53xy: set VIB of operator.
- x is the operator (1-4; last 2 operators only 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 (1-4; last 2 operators only in 4-op mode). a value of 0 means "all operators".
- y is the value.
- 55xy set SUS of operator.

- x is the operator (1-4; last 2 operators only 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 (1-4; last 2 operators only in 4-op mode). a value of 0 means "all operators".
- y determines whether KSR is on.

Yamaha YM2413/OPLL

the YM2413, otherwise known as OPLL, is a cost-reduced FM synthesis sound chip, based on the Yamaha YM3812 (OPL2). thought OPL was downgraded enough? :p

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
- 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

- 11xx: set feedback of channel.
- 12xx: set operator 1 level.
- 13xx: set operator 2 level.
- 16xy: set multiplier of operator.
- x is the operator (1 or 2).
- y is the mutliplier.
- 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.
- 50xy: set AM of operator.
- x is the operator (1-2). a value of 0 means "all operators".
- y determines whether AM is on.
- 51xy set SL of operator.
- x is the operator (1-2). a value of 0 means "all operators".
- y is the value.
- 52xy set RR of operator.
- x is the operator (1-2). a value of 0 means "all operators".
- y is the value.
- 53xy: set VIB of operator.
- x is the operator (1-2). a value of 0 means "all operators".
- y determines whether VIB is on.
- 54xy set KSL of operator.
- x is the operator (1-2). a value of 0 means "all operators".
- y is the value.
- 55xy set EGT of operator.
- x is the operator (1-2). 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 (1-2). a value of 0 means "all operators".
- y determines whether KSR is on.

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 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 mutliplier.
- 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.

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)
was Nintendo's most fearsome rival, completely defeating Sega Mega Drive and defending itself 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 PCM mode.
- **this effect is there for compatibility reasons** - it is otherwise recommended to use Sample type instruments (which automatically enable PCM mode when used).

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...

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.

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.

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 24038 Hz 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, however they cannot be used in Furnace yet. they have been reserved in case this feature is added later. ADPCM samples are limited to 8012 Hz.

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 the length of the echo delay buffer.

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 external PCM data.

effects

none so far.

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.

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.

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 for mostly DefleMask compatibility - it is otherwise recommended to use Sample type instruments.

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.

Ultimately, most of the games for the Game.com ended up being failiures in the eyes of reviewers, thus giving the Game.com a pretty bad reputation. This was one of the reasons that the Game.com only ended up selling at least 300,000 units. For these reasons and more, the Game.com ended up being discontinued in 2000.

However, for its time, it was a pretty competitively priced system. The Gameboy Color was to be released in a year for $79.95, while the Game.com was released for $69.99, and 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, it is not implemented in Furnace as of version 0.6pre4.

effect commands

- `10xx` Set waveform
- `xx` is a value between 0 and 255, that sets the waveform of the channel you place it on.

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.

the original iteration of the SN76489 used in the TI-99/4A computers was clocked at 447 KHz, being able to play as low as 13.670 Hz (A -1). consequentially, pitch accuracy for higher notes is compromised.

on the other hand, the chip was clocked at a much higher speed on Master System and Genesis, which makes it rather poor in the bass range.

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.

Super Nintendo Entertainment System (SNES)/ Super Famicom

the successor to NES to compete with Genesis, packing superior graphics and sample-based audio.

its audio system, developed by Sony, 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, meaning that you can use 2 channels to make a basic FM synth without eating up too much memory.
- 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. this however is not a hardware feature, and might be difficult to implement on real hardware.

effects

- 10xx: set waveform.
- 11xx: toggle noise mode.
- 12xx: toggle echo on this channel.
- 13xx: toggle pitch modulation.

- 14xy: toggle inverting the left or right channels (x: left, y: right).
- 15xx: set envelope mode.
- 0: ADSR.
- 1: gain (direct).
- 2: linear decrement.
- 3: exponential decrement.
- 4: linear increment.
- 5: bent line (inverse log) increment.
- 16xx: set gain (00 to 7F if direct, 00 to 1F otherwise).
- 18xx: enable echo buffer.
- 19xx: set echo delay
- goes from 0 to F.
- 1Axx: set left echo channel volume.
- this is a signed number.
- 00 to 7F for 0 to 127.
- 80 to FF for -128 to -1.
    - setting this to -128 is not recommended as it may cause echo output to overflow and therefore click.
- 1Bxx: set right echo channel volume.
- this is a signed number.
- 00 to 7F for 0 to 127.
- 80 to FF for -128 to -1.
    - setting this to -128 is not recommended as it may cause echo output to overflow and therefore click.
- 1Cxx: set echo feedback.
- this is a signed number.
- 00 to 7F for 0 to 127.
- 80 to FF for -128 to -1.
    - setting this to -128 is not recommended as it may cause echo output to overflow and therefore click.
- 1Dxx: set noise generator frequency (00 to 1F).
- 20xx: set attack (0 to F).
- only in ADSR envelope mode.
- 21xx: set decay (0 to 7).
- only in ADSR envelope mode.
- 22xx: set sustain (0 to 7).
- only in ADSR envelope mode.
- 23xx: set release (00 to 1F).
- only in ADSR envelope mode.
- 30xx: set echo filter coefficient 0.
- 31xx: set echo filter coefficient 1.
- 32xx: set echo filter coefficient 2.
- 33xx: set echo filter coefficient 3.
- 34xx: set echo filter coefficient 4.

- 35xx: set echo filter coefficient 5.
- 36xx: set echo filter coefficient 6.
- 37xx: set echo filter coefficient 7.
- all of these are signed numbers.
- 00 to 7F for 0 to 127.
- 80 to FF for -128 to -1.
- make sure the sum of these is between -128 or 127.
    - failure to comply may result in overflow and therefore clicking.

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 (0 to 7F)
- 13xx: set resonance of filter (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 (0 to FFF)

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.

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. xx may be one of:
- 0: nothing
- 1: buzzy
- 2: low buzzy
- 3: flangy
- 4: square
- 5: square
- 6: pure buzzy
- 7: reedy
- 8: noise
- 9: reedy
- 10: pure buzzy
- 11: nothing
- 12: low square
- 13: low square
- 14: low pure buzzy
- 15: low reedy

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. the following values are accepted:
- 0: pulse
- 1: saw
- 2: triangle
- 3: noise
- 22xx: set duty cycle. xx may go from 0 to 3F.

Commodore VIC-20

the Commodore VIC-20 was Commodore's major attempt at making a personal home computer, and is the percursor 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 (`xx` from `00` to `0F`)

Virtual Boy

a "portable" video game console made by Nintendo in the '90's.

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 (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. it should have a height of 255.
- this is an alternative to setting the modulation wave through the instrument.

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 (0 to 7).
- 17xx: toggle PCM mode.

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 tap preset.
- 12xx: setup sweep period (channel 3 only).
- 0: disable.
- 1-32: enable and set period.
- 13xx: setup sweep amount (channel 3 only).
- 17xx: toggle PCM mode (channel 2 only).

Seta/Allumer X1-010

A sound chip designed by Seta, mainly used in their own arcade hardware from the late 80s to the early 2000s.
It has 2 output channels, but there is no known hardware taking advantage of stereo sound capabilities.
Later hardware paired this with external bankswitching logic, but this isn't emulated yet.
Allumer rebadged it for their own arcade hardware.

It has 16 channels, which can all be switched between PCM sample or wavetable playback mode.
Wavetable playback needs to paired with envelope, similar to AY PSG, but shapes are stored in RAM and as such are user-definable.

In Furnace, this chip can be configured for original arcade mono output or stereo output - it simulates early 'incorrect' emulation on some mono hardware, but it is also based on the assumption that each channel is connected to each output.

Waveform types

This chip supports 2 types of waveforms, needs to be paired to external 8 KB RAM to access these features:

One is a signed 8 bit mono waveform, operated like other wavetable based sound systems.
These are stored at the lower half of RAM at common case.

The other one ("Envelope") is a 4 bit stereo waveform, multiplied with the above and calculates final output, each nibble is used for each output channel.
These are stored at the upper half of RAM at common case.

Both waveforms are 128 bytes (fixed size), freely allocated at each half of RAM except the channel register area: each half can store total 32/31 waveforms at once.
In furnace, you can enable the envelope shape split mode. When it is set, its waveform will be split to the left and right halves for each output. Each max size is 128 bytes, total 256 bytes.

effects

- `10xx`: change wave.
- `11xx`: change envelope shape (also wavetable).
- `17xx`: toggle PCM mode.
- `20xx`: set PCM frequency (1 to FF).
- `22xx`: set envelope mode.
- bit 0 sets whether envelope will affect this channel.
- bit 1 toggles the envelope one-shot mode. when it is set, channel is halted after envelope cycle is finished.
- bit 2 toggles the envelope shape split mode. when it is set, envelope shape will be split to left half and right half.
- 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.

- PCM frequency: `255` step, formula: `step * (Chip clock / 8192)`; 1.95KHz to 498KHz if Chip clock is 16MHz.

Yamaha YM2151

the sound chip powering several arcade boards, the Sharp X1/X68000 and the Commander X16. 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 (page 87)** or **OKI's line of ADPCM chips (page .**

effects

- `10`xx: 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 mutliplier.
- 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.
- `30`xx: enable envelope hard reset.
- this works by inserting a quick release and tiny delay before a new note.
- `50`xy: 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.

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 (1-4).
- y is the mutliplier.
- 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:
- 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. 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.
- `30`xx: enable envelope hard reset.
- this works by inserting a quick release and tiny delay before a new note.
- `50`xy: 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.

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 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 mutliplier.
- 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:
- 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. 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.
- `30`xx: enable envelope hard reset.
- this works by inserting a quick release and tiny delay before a new note.
- `50`xy: 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.

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 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 mutliplier.
- 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. xx may be one of the following:
- 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. 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.
- `30`xx: enable envelope hard reset.
- this works by inserting a quick release and tiny delay before a new note.
- `50`xy: 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.

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 mutliplier.
- 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:
- 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. 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.

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.

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 mutliplier.
- 17xx: enable PCM channel.
- this only works on channel 6.
- 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.

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 0.6 loads 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.

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.

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 as of 0.6 Furnace supports only a Follin/SFX-like engine with 6 channels of narrow pulse wave and click 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!

ROM export technical details

instrument data

TODO

macro data

read length, loop and then release (1 byte).
if it is a 2-byte macro, read a dummy byte.

then read data.

binary command stream

Furnace Command Stream, split version.

```
size | description
-----|---------------------------------
  4  | "FCS\0" format magic
  4  | channel count
 4?? | pointers to channel data
 1?? | preset delays
     | - 16 values
 1?? | speed dial commands
     | - 16 values
 ??? | channel data
```

read command and values (if any).
the list of commands follows.

```
hex | description
----|---------------------------------
 00 | note on: C-(-5)
 01 | note on: C#(-5)
 02 | note on: D-(-5)
 .. | ...
 b1 | note on: A-9
 b2 | note on: A#9
 b3 | note on: B-9
 b4 | note on: null
----|---------------------------------
 b5 | note off
 b6 | note off env
 b7 | env release
 b8 | instrument // (ins, force)
 be | panning // (left, right)
 c0 | pre porta // (inporta, isportaorslide)
```

```
 c2 | vibrato // (speed, depth)
 c3 | vibrato range // (range)
 c4 | vibrato shape // (shape)
 c5 | pitch // (pitch)
 c6 | arpeggio // (note1, note2)
 c7 | volume // (vol)
 c8 | vol slide // (amount, onetick)
 c9 | porta // (target, speed)
 ca | legato // (note)
----|-----------------------------------
 d0 | speed dial command 0
 d1 | speed dial command 1
 .. | ...
 df | speed dial command 15
----|-----------------------------------
 e0 | preset delay 0
 e1 | preset delay 1
 .. | ...
 ef | preset delay 15
----|-----------------------------------
 f7 | full command (command and data follows)
 f8 | go to sub-block (offset follows)
 f9 | return from sub-block
 fa | jump (offset follows)
 fb | set tick rate (4 bytes)
 fc | wait (16-bit)
 fd | wait (8-bit)
 fe | wait one tick
 ff | stop
```

the Furnace file format (.fur)

while Furnace works directly with the .dmf format, I had to create a new format to handle future additions to the program.

this document has the goal of detailing the format.

notice: GitHub's Markdown formatter may break on this file as it doesn't seem to treat tables correctly.

information

files may be zlib-compressed, but Furnace accepts uncompressed files as well.

all numbers are little-endian.

the following fields may be found in "size":
- `f` indicates a floating point number.
- `STR` is a UTF-8 zero-terminated string.
- `???` is an array of variable size.
- `S??` is an array of STRs.
- `1??` is an array of bytes.
- `2??` is an array of shorts.
- `4??` is an array of ints.

the format has changed several times across versions. a `(>=VER)` indicates this field is only present starting from format version VER, and `(<VER)` indicates this field is present only before version VER.

furthermore, an `or reserved` indicates this field is always present, but is reserved when the version condition is not met.

the `size of this block` fields represent the size of a block excluding the `ID` and the aforementioned field.
these fields are 0 in format versions prior to 100 (0.6pre1).

format versions

the format versions are:

- 143: Furnace 0.6pre4
- 142: Furnace dev142
- 141: Furnace Tournament Edition (for intro tune contest)

- 140: Furnace dev140
- 139: Furnace dev139
- 138: Furnace dev138
- 137: Furnace dev137
- 136: Furnace dev136
- 135: Furnace dev135
- 134: Furnace dev134
- 133: Furnace 0.6pre3
- 132: Furnace 0.6pre2
- 131: Furnace dev131
- 130: Furnace dev130
- 129: Furnace dev129
- 128: Furnace dev128
- 127: Furnace dev127
- 126: Furnace dev126
- 125: Furnace dev125
- 124: Furnace dev124
- 123: Furnace dev123
- 122: Furnace dev122
- 121: Furnace dev121
- 120: Furnace dev120
- 119: Furnace dev119
- 118: Furnace dev118
- 117: Furnace dev117
- 116: Furnace 0.6pre1.5
- 115: Furnace dev115
- 114: Furnace dev114
- 113: Furnace dev113
- 112: Furnace dev112
- 111: Furnace dev111
- 110: Furnace dev110
- 109: Furnace dev109
- 108: Furnace dev108
- 107: Furnace dev107
- 106: Furnace dev106
- 105: Furnace dev105
- 104: Furnace dev104
- 103: Furnace dev103
- 102: Furnace 0.6pre1 (dev102)
- 101: Furnace 0.6pre1 (dev101)
- 100: Furnace 0.6pre1
- 99: Furnace dev99
- 98: Furnace dev98
- 97: Furnace dev97

- 96: Furnace dev96
- 95: Furnace dev95
- 94: Furnace dev94
- 93: Furnace dev93
- 92: Furnace dev92
- 91: Furnace dev91
- 90: Furnace dev90
- 89: Furnace dev89
- 88: Furnace dev88
- 87: Furnace dev87
- 86: Furnace dev86
- 85: Furnace dev85
- 84: Furnace dev84
- 83: Furnace dev83
- 82: Furnace dev82
- 81: Furnace dev81
- 80: Furnace dev80
- 79: Furnace dev79
- 78: Furnace dev78
- 77: Furnace dev77
- 76: Furnace dev76
- 75: Furnace dev75/April Fools' 0.6pre0
- 74: Furnace dev74
- 73: Furnace dev73
- 72: Furnace dev72
- 71: Furnace dev71
- 70: Furnace dev70
- 69: Furnace dev69
- 68: Furnace dev68
- 67: Furnace dev67
- 66: Furnace dev66
- 65: Furnace dev65
- 64: Furnace dev64
- 63: Furnace dev63
- 62: Furnace dev62
- 61: Furnace dev61
- 60: Furnace dev60
- 59: Furnace dev59
- 58: Furnace dev58

- 57: Furnace dev57

- 54: Furnace 0.5.8

- 53: Furnace 0.5.7

- 52: Furnace 0.5.7pre4
- 51: Furnace 0.5.7pre3
- 50: Furnace 0.5.7pre2
- 49: Furnace 0.5.7pre1
- 48: Furnace 0.5.6
- 47: Furnace 0.5.6pre1
- 46: Furnace 0.5.5
- 45: Furnace 0.5.5pre3
- 44: Furnace 0.5.5pre2
- 43: Furnace 0.5.5pre1
- 42: Furnace 0.5.4
- 41: Furnace 0.5.3
- 40: Furnace 0.5.2
- 39: Furnace 0.5.2pre3
- 38: Furnace 0.5.2pre2
- 37: Furnace 0.5.2pre1
- 36: Furnace 0.5.1
- 35: Furnace 0.5
- 27: Furnace 0.4.6
- 26: Furnace 0.4.6pre1
- 25: Furnace 0.4.5
- 24: Furnace 0.4.4
- 23: Furnace 0.4.3
- 22: Furnace 0.4.2
- 21: Furnace 0.4.1
- 20: Furnace 0.4
- 19: Furnace 0.4pre3
- 18: Furnace 0.4pre2
- 17: Furnace 0.4pre1
- 16: Furnace 0.3.1
- 15: Furnace 0.3
- 14: Furnace 0.2.2
- 13: Furnace 0.2.1
- 12: Furnace 0.2

header

the header is 32 bytes long.

```
size | description
-----|-----------------------------------
 16  | "-Furnace module-" format magic
  2  | format version
  2  | reserved
```

```
  4  | song info pointer
  8  | reserved
```

song info

```
size | description
-----|-----------------------------------
  4  | "INFO" block ID
  4  | size of this block
  1  | time base (of first song)
  1  | speed 1 (of first song)
  1  | speed 2 (of first song)
  1  | initial arpeggio time (of first song)
  4f | ticks per second (of first song)
     | - 60 is NTSC
     | - 50 is PAL
  2  | pattern length (of first song)
     | - the limit is 256.
  2  | orders length (of first song)
     | - the limit is 256 (>=80) or 127 (<80).
  1  | highlight A (of first song)
  1  | highlight B (of first song)
  2  | instrument count
     | - the limit is 256.
  2  | wavetable count
     | - the limit is 256.
  2  | sample count
     | - the limit is 256.
  4  | pattern count (global)
 32  | list of sound chips
     | - possible soundchips:
     |   - 0x00: end of list
     |   - 0x01: YMU759 - 17 channels
     |   - 0x02: Genesis - 10 channels (compound!)
     |   - 0x03: SMS (SN76489) - 4 channels
     |   - 0x04: Game Boy - 4 channels
     |   - 0x05: PC Engine - 6 channels
     |   - 0x06: NES - 5 channels
     |   - 0x07: C64 (8580) - 3 channels
     |   - 0x08: Arcade (YM2151+SegaPCM) - 13 channels (compound!)
     |   - 0x09: Neo Geo CD (YM2610) - 13 channels
     |   - 0x42: Genesis extended - 13 channels
     |   - 0x43: SMS (SN76489) + OPLL (YM2413) - 13 channels (compound!)
     |   - 0x46: NES + VRC7 - 11 channels (compound!)
     |   - 0x47: C64 (6581) - 3 channels
     |   - 0x49: Neo Geo CD extended - 16 channels
     |   - 0x80: AY-3-8910 - 3 channels
     |   - 0x81: Amiga - 4 channels
     |   - 0x82: YM2151 alone - 8 channels
     |   - 0x83: YM2612 alone - 6 channels
     |   - 0x84: TIA - 2 channels
     |   - 0x85: VIC-20 - 4 channels
     |   - 0x86: PET - 1 channel
     |   - 0x87: SNES - 8 channels
```

```
|    - 0x88: VRC6 - 3 channels
|    - 0x89: OPLL (YM2413) - 9 channels
|    - 0x8a: FDS - 1 channel
|    - 0x8b: MMC5 - 3 channels
|    - 0x8c: Namco 163 - 8 channels
|    - 0x8d: YM2203 - 6 channels
|    - 0x8e: YM2608 - 16 channels
|    - 0x8f: OPL (YM3526) - 9 channels
|    - 0x90: OPL2 (YM3812) - 9 channels
|    - 0x91: OPL3 (YMF262) - 18 channels
|    - 0x92: MultiPCM - 28 channels
|    - 0x93: Intel 8253 (beeper) - 1 channel
|    - 0x94: POKEY - 4 channels
|    - 0x95: RF5C68 - 8 channels
|    - 0x96: WonderSwan - 4 channels
|    - 0x97: Philips SAA1099 - 6 channels
|    - 0x98: OPZ (YM2414) - 8 channels
|    - 0x99: Pokémon Mini - 1 channel
|    - 0x9a: AY8930 - 3 channels
|    - 0x9b: SegaPCM - 16 channels
|    - 0x9c: Virtual Boy - 6 channels
|    - 0x9d: VRC7 - 6 channels
|    - 0x9e: YM2610B - 16 channels
|    - 0x9f: ZX Spectrum (beeper, tildearrow engine) - 6 channels
|    - 0xa0: YM2612 extended - 9 channels
|    - 0xa1: Konami SCC - 5 channels
|    - 0xa2: OPL drums (YM3526) - 11 channels
|    - 0xa3: OPL2 drums (YM3812) - 11 channels
|    - 0xa4: OPL3 drums (YMF262) - 20 channels
|    - 0xa5: Neo Geo (YM2610) - 14 channels
|    - 0xa6: Neo Geo extended (YM2610) - 17 channels
|    - 0xa7: OPLL drums (YM2413) - 11 channels
|    - 0xa8: Atari Lynx - 4 channels
|    - 0xa9: SegaPCM (for DefleMask compatibility) - 5 channels
|    - 0xaa: MSM6295 - 4 channels
|    - 0xab: MSM6258 - 1 channel
|    - 0xac: Commander X16 (VERA) - 17 channels
|    - 0xad: Bubble System WSG - 2 channels
|    - 0xae: OPL4 (YMF278B) - 42 channels
|    - 0xaf: OPL4 drums (YMF278B) - 44 channels
|    - 0xb0: Seta/Allumer X1-010 - 16 channels
|    - 0xb1: Ensoniq ES5506 - 32 channels
|    - 0xb2: Yamaha Y8950 - 10 channels
|    - 0xb3: Yamaha Y8950 drums - 12 channels
|    - 0xb4: Konami SCC+ - 5 channels
|    - 0xb5: tildearrow Sound Unit - 8 channels
|    - 0xb6: YM2203 extended - 9 channels
|    - 0xb7: YM2608 extended - 19 channels
|    - 0xb8: YMZ280B - 8 channels
|    - 0xb9: Namco WSG - 3 channels
|    - 0xba: Namco 15xx - 8 channels
|    - 0xbb: Namco CUS30 - 8 channels
|    - 0xbc: MSM5232 - 8 channels
|    - 0xbd: YM2612 extra features extended - 11 channels
|    - 0xbe: YM2612 extra features - 7 channels
```

```
    |   - 0xbf: T6W28 - 4 channels
    |   - 0xc0: PCM DAC - 1 channel
    |   - 0xc1: YM2612 CSM - 10 channels
    |   - 0xc2: Neo Geo CSM (YM2610) - 18 channels
    |   - 0xc3: YM2203 CSM - 10 channels
    |   - 0xc4: YM2608 CSM - 20 channels
    |   - 0xc5: YM2610B CSM - 20 channels
    |   - 0xc6: K007232 - 2 channels
    |   - 0xc7: GA20 - 4 channels
    |   - 0xc8: SM8521 - 3 channels
    |   - 0xc9: M114S - 16 channels
    |   - 0xca: ZX Spectrum (beeper, QuadTone engine) - 5 channels
    |   - 0xcb: Casio PV-1000 - 3 channels
    |   - 0xde: YM2610B extended - 19 channels
    |   - 0xe0: QSound - 19 channels
    |   - 0xfc: Pong - 1 channel
    |   - 0xfd: Dummy System - 8 channels
    |   - 0xfe: reserved for development
    |   - 0xff: reserved for development
    | - (compound!) means that the system is composed of two or more chips,
    |   and has to be flattened.
32  | sound chip volumes (<135) or reserved
    | - signed char, 64=1.0, 127=~2.0
    | - as of version 135 these fields only exist for compatibility reasons.
32  | sound chip panning (<135) or reserved
    | - signed char, -128=left, 127=right
    | - as of version 135 these fields only exist for compatibility reasons.
128 | sound chip flag pointers (>=119) or sound chip flags
    | - before 118, these were 32-bit flags.
    | - for conversion details, see the "converting from old flags" section.
STR | song name
STR | song author
 4f | A-4 tuning
 1  | limit slides (>=36) or reserved
 1  | linear pitch (>=36) or reserved
    | - 0: non-linear
    | - 1: only pitch change (04xy/E5xx) linear
    | - 2: full linear (>=94)
 1  | loop modality (>=36) or reserved
 1  | proper noise layout (>=42) or reserved
 1  | wave duty is volume (>=42) or reserved
 1  | reset macro on porta (>=45) or reserved
 1  | legacy volume slides (>=45) or reserved
 1  | compatible arpeggio (>=45) or reserved
 1  | note off resets slides (>=45) or reserved
 1  | target resets slides (>=45) or reserved
 1  | arpeggio inhibits portamento (>=47) or reserved
 1  | wack algorithm macro (>=47) or reserved
 1  | broken shortcut slides (>=49) or reserved
 1  | ignore duplicate slides (>=50) or reserved
 1  | stop portamento on note off (>=62) or reserved
 1  | continuous vibrato (>=62) or reserved
 1  | broken DAC mode (>=64) or reserved
 1  | one tick cut (>=65) or reserved
 1  | instrument change allowed during porta (>=66) or reserved
```

```
 1  | reset note base on arpeggio effect stop (0000) (>=69) or reserved
4?? | pointers to instruments
4?? | pointers to wavetables
4?? | pointers to samples
4?? | pointers to patterns
??? | orders (of first song)
    | - a table of bytes
    | - size=channels*ordLen
    | - read orders then channels
    | - the maximum value of a cell is FF (>=80) or 7F (<80).
??? | effect columns (of first song)
    | - size=channels
1?? | channel hide status (of first song)
    | - size=channels
1?? | channel collapse status (of first song)
    | - size=channels
S?? | channel names (of first song)
    | - a list of channelCount C strings
S?? | channel short names (of first song)
    | - same as above
STR | song comment
 4f | master volume, 1.0f=100% (>=59)
    | this is 2.0f for modules before 59
--- | **extended compatibility flags** (>=70)
 1  | broken speed selection
 1  | no slides on first tick (>=71) or reserved
 1  | next row reset arp pos (>=71) or reserved
 1  | ignore jump at end (>=71) or reserved
 1  | buggy portamento after slide (>=72) or reserved
 1  | new ins affects envelope (Game Boy) (>=72) or reserved
 1  | ExtCh channel state is shared (>=78) or reserved
 1  | ignore DAC mode change outside of intended channel (>=83) or reserved
 1  | E1xy and E2xy also take priority over Slide00 (>=83) or reserved
 1  | new Sega PCM (with macros and proper vol/pan) (>=84) or reserved
 1  | weird f-num/block-based chip pitch slides (>=85) or reserved
 1  | SN duty macro always resets phase (>=86) or reserved
 1  | pitch macro is linear (>=90) or reserved
 1  | pitch slide speed in full linear pitch mode (>=94) or reserved
 1  | old octave boundary behavior (>=97) or reserved
 1  | disable OPN2 DAC volume control (>=98) or reserved
 1  | new volume scaling strategy (>=99) or reserved
 1  | volume macro still applies after end (>=99) or reserved
 1  | broken outVol (>=99) or reserved
 1  | E1xy and E2xy stop on same note (>=100) or reserved
 1  | broken initial position of porta after arp (>=101) or reserved
 1  | SN periods under 8 are treated as 1 (>=108) or reserved
 1  | cut/delay effect policy (>=110) or reserved
 1  | 0B/0D effect treatment (>=113) or reserved
 1  | automatic system name detection (>=115) or reserved
    | - this one isn't a compatibility flag, but it's here for convenience...
 1  | disable sample macro (>=117) or reserved
 1  | broken outVol episode 2 (>=121) or reserved
 1  | old arpeggio strategy (>=130) or reserved
--- | **virtual tempo data**
 2  | virtual tempo numerator of first song (>=96) or reserved
```

```
 2  | virtual tempo denominator of first song (>=96) or reserved
--- | **additional subsongs** (>=95)
STR | first subsong name
STR | first subsong comment
 1  | number of additional subsongs
 3  | reserved
4?? | pointers to subsong data
--- | **additional metadata** (>=103)
STR | system name
STR | album/category/game name
STR | song name (Japanese)
STR | song author (Japanese)
STR | system name (Japanese)
STR | album/category/game name (Japanese)
--- | **extra chip output settings (× chipCount)** (>=135)
 4f | chip volume
 4f | chip panning
 4f | chip front/rear balance
--- | **patchbay** (>=135)
 4  | patchbay connection count
4?? | patchbay
    | - see next section for more details.
 1  | automatic patchbay (>=136)
--- | **a couple more compat flags** (>=138)
 1  | broken portamento during legato
 7  | reserved
--- | **speed pattern of first song** (>=139)
 1  | length of speed pattern (fail if this is lower than 0 or higher than 16)
16  | speed pattern (this overrides speed 1 and speed 2 settings)
--- | **groove list** (>=139)
 1  | number of entries
??? | groove entries. the format is:
    | - 1 byte: length of groove
    | - 16 bytes: groove pattern
```

patchbay

Furnace dev135 adds a "patchbay" which allows for arbitrary connection of chip outputs to system outputs.
it eventually will allow connecting outputs to effects and so on.

a connection is represented as an unsigned int in the following format:

- bit 16-31: source port
- bit 0-15: destination port

a port is in the following format (hexadecimal): xxxy

- xxx (bit 4 to 15) represents a portset.
- y (bit 0 to 3) is the port in that portset.

reserved input portsets:
- 000: system outputs
- FFF: "null" portset

reserved output portsets:
- 000 through 01F: chip outputs
- FFD: wave/sample preview
- FFE: metronome
- FFF: "null" portset

subsong

from version 95 onwards, Furnace supports storing multiple songs on a single file.
the way it's currently done is really weird, but it provides for some backwards compatibility (previous versions will only load the first subsong which is already defined in the INFO block).

```
size | description
-----|-----------------------------------
  4  | "SONG" block ID
  4  | size of this block
  1  | time base
  1  | speed 1
  1  | speed 2
  1  | initial arpeggio time
  4f | ticks per second
     | - 60 is NTSC
     | - 50 is PAL
  2  | pattern length
     | - the limit is 256.
  2  | orders length
     | - the limit is 256.
  1  | highlight A
  1  | highlight B
  2  | virtual tempo numerator
  2  | virtual tempo denominator
 STR | subsong name
 STR | subsong comment
 ??? | orders
     | - a table of bytes
     | - size=channels*ordLen
     | - read orders then channels
     | - the maximum value of a cell is FF.
 ??? | effect columns
     | - size=channels
 1?? | channel hide status
     | - size=channels
 1?? | channel collapse status
     | - size=channels
 S?? | channel names
```

```
     | - a list of channelCount C strings
 S?? | channel short names
     | - same as above
 --- | **speed pattern** (>=139)
  1  | length of speed pattern (fail if this is lower than 0 or higher than 16)
 16  | speed pattern (this overrides speed 1 and speed 2 settings)
```

chip flags

```
size | description
-----|-----------------------------------
  4  | "FLAG" block ID
  4  | size of this block
 STR | data
```

flags are stored in text (`key=value`) format. for example:

```
clock=4000000
stereo=true
```

instrument (>=127)

Furnace dev127 and higher use the new instrument format.

```
size | description
-----|-----------------------------------
  4  | "INS2" block ID
  4  | size of this block
  2  | format version
  2  | instrument type
 ??? | features...
```

see **newIns.md** **(page 161)** for more information.

old instrument (<127)

notes:

- the entire instrument is stored, regardless of instrument type.
- the macro range varies depending on the instrument type.
- "macro open" indicates whether the macro is collapsed or not in the instrument editor.
- as of format version 120, bit 1-2 indicates macro mode:
  - 0: sequence (normal)
  - 1: ADSR
  - 2: LFO
- see sub-section for information on how to interpret parameters.
- FM operator order is:

- 1/3/2/4 (internal order) for OPN, OPM, OPZ and OPL 4-op
- 1/2/?/? (? = unused) for OPL 2-op and OPLL
- meaning of extended macros varies depending on instrument type.
- meaning of panning macros varies depending on instrument type:
- for hard-panned chips (e.g. FM and Game Boy): left panning is 2-bit panning macro (left/right)
- otherwise both left and right panning macros are used

```
size | description
-----|-----------------------------------
  4  | "INST" block ID
  4  | size of this block
  2  | format version (see header)
  1  | instrument type
     | - 0: SN76489/standard
     | - 1: FM (OPN)
     | - 2: Game Boy
     | - 3: C64
     | - 4: Amiga/sample
     | - 5: PC Engine
     | - 6: AY-3-8910
     | - 7: AY8930
     | - 8: TIA
     | - 9: SAA1099
     | - 10: VIC
     | - 11: PET
     | - 12: VRC6
     | - 13: OPLL
     | - 14: OPL
     | - 15: FDS
     | - 16: Virtual Boy
     | - 17: Namco 163
     | - 18: SCC
     | - 19: OPZ
     | - 20: POKEY
     | - 21: PC Speaker
     | - 22: WonderSwan
     | - 23: Lynx
     | - 24: VERA
     | - 25: X1-010
     | - 26: VRC6 (saw)
     | - 27: ES5506
     | - 28: MultiPCM
     | - 29: SNES
     | - 30: Sound Unit
     | - 31: Namco WSG
     | - 32: OPL (drums)
     | - 33: FM (OPM)
     | - 34: NES
     | - 35: MSM6258
     | - 36: MSM6295
     | - 37: ADPCM-A
     | - 38: ADPCM-B
     | - 39: SegaPCM
```

```
    | - 40: QSound
    | - 41: YMZ280B
    | - 42: RF5C68
    | - 43: MSM5232
    | - 44: T6W28
 1  | reserved
STR | instrument name
--- | **FM instrument data**
 1  | alg (SUS on OPLL)
 1  | feedback
 1  | fms (DC on OPLL)
 1  | ams (DM on OPLL)
 1  | operator count
    | - this is either 2 or 4, and is ignored on non-OPL systems.
    | - always read 4 ops regardless of this value.
 1  | OPLL preset (>=60) or reserved
    | - 0: custom
    | - 1-15: pre-defined patches
    | - 16: drums (compatibility only!)
 2  | reserved
--- | **FM operator data** × 4
 1  | am
 1  | ar
 1  | dr
 1  | mult
 1  | rr
 1  | sl
 1  | tl
 1  | dt2
 1  | rs
 1  | dt
 1  | d2r
 1  | ssgEnv
    | - bit 4: on (EG-S on OPLL)
    | - bit 0-3: envelope type
 1  | dam (for YMU759 compat; REV on OPZ)
 1  | dvb (for YMU759 compat; FINE on OPZ)
 1  | egt (for YMU759 compat; FixedFreq on OPZ)
 1  | ksl (EGShift on OPZ)
 1  | sus
 1  | vib
 1  | ws
 1  | ksr
 1  | operator enabled (>=114) or reserved
 1  | KVS mode (>=115) or reserved
    | - 0: off
    | - 1: on
    | - 2: auto (depending on alg)
10  | reserved
--- | **Game Boy instrument data**
 1  | volume
 1  | direction
 1  | length
 1  | sound length
--- | **C64 instrument data**
```

```
 1  | triangle
 1  | saw
 1  | pulse
 1  | noise
 1  | attack
 1  | decay
 1  | sustain
 1  | release
 2  | duty
 1  | ring mod
 1  | osc sync
 1  | to filter
 1  | init filter
 1  | vol macro is cutoff
 1  | resonance
 1  | low pass
 1  | band pass
 1  | high pass
 1  | channel 3 off
 2  | cutoff
 1  | duty macro is absolute
 1  | filter macro is absolute
---- | **Amiga instrument data**
 2  | initial sample
 1  | mode (>=82) or reserved
    | - 0: sample
    | - 1: wavetable
 1  | wavetable length (-1) (>=82) or reserved
12  | reserved
---- | **standard instrument data**
 4  | volume macro length
 4  | arp macro length
 4  | duty macro length
 4  | wave macro length
 4  | pitch macro length (>=17)
 4  | extra 1 macro length (>=17)
 4  | extra 2 macro length (>=17)
 4  | extra 3 macro length (>=17)
 4  | volume macro loop
 4  | arp macro loop
 4  | duty macro loop
 4  | wave macro loop
 4  | pitch macro loop (>=17)
 4  | extra 1 macro loop (>=17)
 4  | extra 2 macro loop (>=17)
 4  | extra 3 macro loop (>=17)
 1  | arp macro mode (<112) or reserved
    | - treat this value in a special way.
    | - before version 112, this byte indicates whether the arp macro mode is fixed or
    | - from that version onwards, the fixed mode is part of the macro values.
    | - to convert a <112 macro mode to a modern one, do the following:
    |   - is the macro mode set to fixed?
    |     - if yes, then:
    |       - set bit 30 of all arp macro values (this is the fixed mode bit)
    |       - does the macro loop?
```

```
     |         - if yes, then do nothing else
     |         - if no, then add one to the macro length, and set the last macro value
     |     - if no, then do nothing
  1  | reserved (>=17) or volume macro height (>=15) or reserved
  1  | reserved (>=17) or duty macro height (>=15) or reserved
  1  | reserved (>=17) or wave macro height (>=15) or reserved
 4?? | volume macro
     | - before version 87, if this is the C64 relative cutoff macro, its values were s
 4?? | arp macro
     | - before version 31, this macro's values were stored offset by 12.
     | - from version 112 onward, bit 30 of a value indicates fixed mode.
 4?? | duty macro
     | - before version 87, if this is the C64 relative duty macro, its values were sto
 4?? | wave macro
 4?? | pitch macro (>=17)
 4?? | extra 1 macro (>=17)
 4?? | extra 2 macro (>=17)
 4?? | extra 3 macro (>=17)
  4  | alg macro length (>=29)
  4  | fb macro length (>=29)
  4  | fms macro length (>=29)
  4  | ams macro length (>=29)
  4  | alg macro loop (>=29)
  4  | fb macro loop (>=29)
  4  | fms macro loop (>=29)
  4  | ams macro loop (>=29)
  1  | volume macro open (>=29)
  1  | arp macro open (>=29)
  1  | duty macro open (>=29)
  1  | wave macro open (>=29)
  1  | pitch macro open (>=29)
  1  | extra 1 macro open (>=29)
  1  | extra 2 macro open (>=29)
  1  | extra 3 macro open (>=29)
  1  | alg macro open (>=29)
  1  | fb macro open (>=29)
  1  | fms macro open (>=29)
  1  | ams macro open (>=29)
 4?? | alg macro (>=29)
 4?? | fb macro (>=29)
 4?? | fms macro (>=29)
 4?? | ams macro (>=29)
 --- | **operator macro headers** × 4 (>=29)
  4  | AM macro length
  4  | AR macro length
  4  | DR macro length
  4  | MULT macro length
  4  | RR macro length
  4  | SL macro length
  4  | TL macro length
  4  | DT2 macro length
  4  | RS macro length
  4  | DT macro length
  4  | D2R macro length
  4  | SSG-EG macro length
```

```
 4  | AM macro loop
 4  | AR macro loop
 4  | DR macro loop
 4  | MULT macro loop
 4  | RR macro loop
 4  | SL macro loop
 4  | TL macro loop
 4  | DT2 macro loop
 4  | RS macro loop
 4  | DT macro loop
 4  | D2R macro loop
 4  | SSG-EG macro loop
 1  | AM macro open
 1  | AR macro open
 1  | DR macro open
 1  | MULT macro open
 1  | RR macro open
 1  | SL macro open
 1  | TL macro open
 1  | DT2 macro open
 1  | RS macro open
 1  | DT macro open
 1  | D2R macro open
 1  | SSG-EG macro open
---- | **operator macros** × 4 (>=29)
1?? | AM macro
1?? | AR macro
1?? | DR macro
1?? | MULT macro
1?? | RR macro
1?? | SL macro
1?? | TL macro
1?? | DT2 macro
1?? | RS macro
1?? | DT macro
1?? | D2R macro
1?? | SSG-EG macro
---- | **release points** (>=44)
 4  | volume macro release
 4  | arp macro release
 4  | duty macro release
 4  | wave macro release
 4  | pitch macro release
 4  | extra 1 macro release
 4  | extra 2 macro release
 4  | extra 3 macro release
 4  | alg macro release
 4  | fb macro release
 4  | fms macro release
 4  | ams macro release
---- | **operator release points** × 4 (>=44)
 4  | AM macro release
 4  | AR macro release
 4  | DR macro release
 4  | MULT macro release
```

```
 4  | RR macro release
 4  | SL macro release
 4  | TL macro release
 4  | DT2 macro release
 4  | RS macro release
 4  | DT macro release
 4  | D2R macro release
 4  | SSG-EG macro release
--- | **extended op macro headers** × 4 (>=61)
 4  | DAM macro length
 4  | DVB macro length
 4  | EGT macro length
 4  | KSL macro length
 4  | SUS macro length
 4  | VIB macro length
 4  | WS macro length
 4  | KSR macro length
 4  | DAM macro loop
 4  | DVB macro loop
 4  | EGT macro loop
 4  | KSL macro loop
 4  | SUS macro loop
 4  | VIB macro loop
 4  | WS macro loop
 4  | KSR macro loop
 4  | DAM macro release
 4  | DVB macro release
 4  | EGT macro release
 4  | KSL macro release
 4  | SUS macro release
 4  | VIB macro release
 4  | WS macro release
 4  | KSR macro release
 1  | DAM macro open
 1  | DVB macro open
 1  | EGT macro open
 1  | KSL macro open
 1  | SUS macro open
 1  | VIB macro open
 1  | WS macro open
 1  | KSR macro open
--- | **extended op macros** × 4 (>=61)
1?? | DAM macro
1?? | DVB macro
1?? | EGT macro
1?? | KSL macro
1?? | SUS macro
1?? | VIB macro
1?? | WS macro
1?? | KSR macro
--- | **OPL drums mode data** (>=63)
 1  | fixed frequency mode
 1  | reserved
 2  | kick frequency
 2  | snare/hi-hat frequency
```

```
 2  | tom/top frequency
--- | **Sample instrument extra data** (>=67)
 1  | use note map
    | - only read the following two data structures if this is true!
4?? | note frequency × 120
    | - 480 bytes
2?? | note sample × 120
    | - 240 bytes
--- | **Namco 163 data** (>=73)
 4  | initial waveform
 1  | wave position
 1  | wave length
 1  | wave mode:
    | - bit 1: update on change
    | - bit 0: load on playback
 1  | reserved
--- | **even more macros** (>=76)
 4  | left panning macro length
 4  | right panning macro length
 4  | phase reset macro length
 4  | extra 4 macro length
 4  | extra 5 macro length
 4  | extra 6 macro length
 4  | extra 7 macro length
 4  | extra 8 macro length
 4  | left panning macro loop
 4  | right panning macro loop
 4  | phase reset macro loop
 4  | extra 4 macro loop
 4  | extra 5 macro loop
 4  | extra 6 macro loop
 4  | extra 7 macro loop
 4  | extra 8 macro loop
 4  | left panning macro release
 4  | right panning macro release
 4  | phase reset macro release
 4  | extra 4 macro release
 4  | extra 5 macro release
 4  | extra 6 macro release
 4  | extra 7 macro release
 4  | extra 8 macro release
 1  | left panning macro open
 1  | right panning macro open
 1  | phase reset macro open
 1  | extra 4 macro open
 1  | extra 5 macro open
 1  | extra 6 macro open
 1  | extra 7 macro open
 1  | extra 8 macro open
--- | **even more macro data** (>=76)
4?? | left panning macro
4?? | right panning macro
4?? | phase reset macro
4?? | extra 4 macro
4?? | extra 5 macro
```

```
4?? | extra 6 macro
4?? | extra 7 macro
4?? | extra 8 macro
--- | **FDS instrument data** (>=76)
 4  | modulation speed
 4  | modulation depth
 1  | init modulation table with first wave
 3  | reserved
32  | modulation table
--- | **OPZ instrument extra data** (>=77)
 1  | fms2
 1  | ams2
--- | **wavetable synth data** (>=79)
 4  | first wave
 4  | second wave
 1  | rate divider
 1  | effect
    | - bit 7: single or dual effect
 1  | enabled
 1  | global
 1  | speed (+1)
 1  | parameter 1
 1  | parameter 2
 1  | parameter 3
 1  | parameter 4
--- | **additional macro mode flags** (>=84)
 1  | volume macro mode
 1  | duty macro mode
 1  | wave macro mode
 1  | pitch macro mode
 1  | extra 1 macro mode
 1  | extra 2 macro mode
 1  | extra 3 macro mode
 1  | alg macro mode
 1  | fb macro mode
 1  | fms macro mode
 1  | ams macro mode
 1  | left panning macro mode
 1  | right panning macro mode
 1  | phase reset macro mode
 1  | extra 4 macro mode
 1  | extra 5 macro mode
 1  | extra 6 macro mode
 1  | extra 7 macro mode
 1  | extra 8 macro mode
--- | **extra C64 data** (>=89)
 1  | don't test/gate before new note
--- | **MultiPCM data** (>=93)
 1  | attack rate
 1  | decay 1 rate
 1  | decay level
 1  | decay 2 rate
 1  | release rate
 1  | rate correction
 1  | lfo rate
```

```
 1  | vib depth
 1  | am depth
23  | reserved
--- | **Sound Unit data** (>=104)
 1  | use sample
 1  | switch roles of phase reset timer and frequency
--- | **Game Boy envelope sequence** (>=105)
 1  | length
??? | hardware sequence data
    | size is length*3:
    | 1 byte: command
    | - 0: set envelope
    | - 1: set sweep
    | - 2: wait
    | - 3: wait for release
    | - 4: loop
    | - 5: loop until release
    | 2 bytes: data
    | - for set envelope:
    |   - 1 byte: parameter
    |     - bit 4-7: volume
    |     - bit 3: direction
    |     - bit 0-2: length
    |   - 1 byte: sound length
    | - for set sweep:
    |   - 1 byte: parameter
    |     - bit 4-6: length
    |     - bit 3: direction
    |     - bit 0-2: shift
    |   - 1 byte: nothing
    | - for wait:
    |   - 1 byte: length (in ticks)
    |   - 1 byte: nothing
    | - for wait for release:
    |   - 2 bytes: nothing
    | - for loop/loop until release:
    |   - 2 bytes: position
--- | **Game Boy extra flags** (>=106)
 1  | use software envelope
 1  | always init hard env on new note
--- | **ES5506 data** (>=107)
 1  | filter mode
    | - 0: HPK2_HPK2
    | - 1: HPK2_LPK1
    | - 2: LPK2_LPK2
    | - 3: LPK2_LPK1
 2  | K1
 2  | K2
 2  | envelope count
 1  | left volume ramp
 1  | right volume ramp
 1  | K1 ramp
 1  | K2 ramp
 1  | K1 slow
 1  | K2 slow
```

```
---  | **SNES data** (>=109)
 1   | use envelope
 1   | gain mode
 1   | gain
 1   | attack
 1   | decay
 1   | sustain
     | - bit 3: sustain mode (>=118)
 1   | release
---  | **macro speeds/delays** (>=111)
 1   | volume macro speed
 1   | arp macro speed
 1   | duty macro speed
 1   | wave macro speed
 1   | pitch macro speed
 1   | extra 1 macro speed
 1   | extra 2 macro speed
 1   | extra 3 macro speed
 1   | alg macro speed
 1   | fb macro speed
 1   | fms macro speed
 1   | ams macro speed
 1   | left panning macro speed
 1   | right panning macro speed
 1   | phase reset macro speed
 1   | extra 4 macro speed
 1   | extra 5 macro speed
 1   | extra 6 macro speed
 1   | extra 7 macro speed
 1   | extra 8 macro speed
 1   | volume macro delay
 1   | arp macro delay
 1   | duty macro delay
 1   | wave macro delay
 1   | pitch macro delay
 1   | extra 1 macro delay
 1   | extra 2 macro delay
 1   | extra 3 macro delay
 1   | alg macro delay
 1   | fb macro delay
 1   | fms macro delay
 1   | ams macro delay
 1   | left panning macro delay
 1   | right panning macro delay
 1   | phase reset macro delay
 1   | extra 4 macro delay
 1   | extra 5 macro delay
 1   | extra 6 macro delay
 1   | extra 7 macro delay
 1   | extra 8 macro delay
---  | **operator macro speeds/delay** × 4 (>=111)
 1   | AM macro speed
 1   | AR macro speed
 1   | DR macro speed
 1   | MULT macro speed
```

```
1  | RR macro speed
1  | SL macro speed
1  | TL macro speed
1  | DT2 macro speed
1  | RS macro speed
1  | DT macro speed
1  | D2R macro speed
1  | SSG-EG macro speed
1  | DAM macro speed
1  | DVB macro speed
1  | EGT macro speed
1  | KSL macro speed
1  | SUS macro speed
1  | VIB macro speed
1  | WS macro speed
1  | KSR macro speed
1  | AM macro delay
1  | AR macro delay
1  | DR macro delay
1  | MULT macro delay
1  | RR macro delay
1  | SL macro delay
1  | TL macro delay
1  | DT2 macro delay
1  | RS macro delay
1  | DT macro delay
1  | D2R macro delay
1  | SSG-EG macro delay
1  | DAM macro delay
1  | DVB macro delay
1  | EGT macro delay
1  | KSL macro delay
1  | SUS macro delay
1  | VIB macro delay
1  | WS macro delay
1  | KSR macro delay
```

interpreting macro mode values

- sequence (normal): I think this is obvious...
- ADSR:
- `val[0]`: bottom
- `val[1]`: top
- `val[2]`: attack
- `val[3]`: hold time
- `val[4]`: decay
- `val[5]`: sustain level
- `val[6]`: sustain hold time
- `val[7]`: decay 2
- `val[8]`: release
- LFO:

- `val[11]`: speed
- `val[12]`: waveform
  - 0: triangle
  - 1: saw
  - 2: pulse
- `val[13]`: phase
- `val[14]`: loop
- `val[15]`: global (not sure how will I implement this)

wavetable

```
size | description
-----|-----------------------------------
  4  | "WAVE" block ID
  4  | size of this block
 STR | wavetable name
  4  | wavetable width
  4  | reserved
  4  | wavetable height
 4?? | wavetable data
```

sample (>=102)

this is the new sample storage format used in Furnace dev102 and higher.

```
size | description
-----|-----------------------------------
  4  | "SMP2" block ID
  4  | size of this block
 STR | sample name
  4  | length
  4  | compatibility rate
  4  | C-4 rate
  1  | depth
     | - 0: ZX Spectrum overlay drum (1-bit)
     | - 1: 1-bit NES DPCM (1-bit)
     | - 3: YMZ ADPCM
     | - 4: QSound ADPCM
     | - 5: ADPCM-A
     | - 6: ADPCM-B
     | - 8: 8-bit PCM
     | - 9: BRR (SNES)
     | - 10: VOX
     | - 16: 16-bit PCM
  1  | loop direction (>=123) or reserved
     | - 0: forward
     | - 1: backward
     | - 2: ping-pong
  1  | flags (>=129) or reserved
     | - 0: BRR emphasis
  1  | reserved
```

```
  4  | loop start
     | - -1 means no loop
  4  | loop end
     | - -1 means no loop
 16  | sample presence bitfields
     | - for future use.
     | - indicates whether the sample should be present in the memory of a system.
     | - read 4 32-bit numbers (for 4 memory banks per system, e.g. YM2610
     |   does ADPCM-A and ADPCM-B on separate memory banks).
 ??? | sample data
     | - size is length
```

old sample (<102)

this format is present when saving using previous Furnace versions.

```
size | description
-----|-----------------------------------
  4  | "SMPL" block ID
  4  | size of this block
 STR | sample name
  4  | length
  4  | compatibility rate
  2  | volume (<58) or reserved
  2  | pitch (<58) or reserved
  1  | depth
     | - 0: ZX Spectrum overlay drum (1-bit)
     | - 1: 1-bit NES DPCM (1-bit)
     | - 3: YMZ ADPCM
     | - 4: QSound ADPCM
     | - 5: ADPCM-A
     | - 6: ADPCM-B
     | - 8: 8-bit PCM
     | - 9: BRR (SNES)
     | - 10: VOX
     | - 16: 16-bit PCM
  1  | reserved
  2  | C-4 rate (>=32) or reserved
  4  | loop point (>=19) or reserved
     | - -1 means no loop
 ??? | sample data
     | - version<58 size is length*2
     | - version>=58 size is length
```

pattern

```
size | description
-----|-----------------------------------
  4  | "PATR" block ID
  4  | size of this block
  2  | channel
  2  | pattern index
  2  | subsong (>=95) or reserved
```

```
 2  | reserved
???? | pattern data
     | - size: rows*(4+effectColumns*2)*2
     | - read shorts in this order:
     |   - note
     |     - 0: empty/invalid
     |     - 1: C#
     |     - 2: D
     |     - 3: D#
     |     - 4: E
     |     - 5: F
     |     - 6: F#
     |     - 7: G
     |     - 8: G#
     |     - 9: A
     |     - 10: A#
     |     - 11: B
     |     - 12: C (of next octave)
     |       - this is actually a leftover of the .dmf format.
     |     - 100: note off
     |     - 100: note release
     |     - 100: macro release
     |   - octave
     |     - this is an signed char stored in a short.
     |     - therefore octave value 255 is actually octave -1.
     |     - yep, another leftover of the .dmf format...
     |   - instrument
     |   - volume
     |   - effect and effect data (× effect columns)
     | - for note/octave, if both values are 0 then it means empty.
     | - for instrument, volume, effect and effect data, a value of -1 means empty.
 STR | pattern name (>=51)
```

the Furnace instrument format (.fui)

the instrument format is pretty similar to the file format, but it also stores wavetables and samples used by the instrument.

```
size | description
-----|---------------------------------
 16  | "-Furnace instr.-" format magic
  2  | format version
  2  | reserved
  4  | pointer to instrument data
  2  | wavetable count
  2  | sample count
  4  | reserved
 4?? | pointers to wavetables
 4?? | pointers to samples
```

instrument data follows.

the Furnace wavetable format (.fuw)

similar to the instrument format...

```
size | description
-----|-----------------------------------
 16  | "-Furnace waveta-" format magic
  2  | format version
  2  | reserved
```

wavetable data follows.

converting from old flags

prior to format version 119, chip flags were stored as a 32-bit integer.
this section will help you understand the old flag format.

chips which aren't on this list don't have any flags.

0x02: Genesis (COMPOUND) and 0x42: Genesis extended (COMPOUND)

- bit 31: ladderEffect (bool)
- bit 0-30: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 8MHz
- 3: Firecore (`COLOR_NTSC*12/7`)
- 4: System 32 (`COLOR_NTSC*9/4`)
- only 0 and 1 apply to the SN part as well.

0x03: SMS (SN76489)

- flags AND 0xff03: clockSel (int)
- 0x0000: NTSC (becomes 0)
- 0x0001: PAL (becomes 1)
- 0x0002: 4MHz (becomes 2)
- 0x0003: half NTSC (becomes 3)
- 0x0100: 3MHz (becomes 4)
- 0x0101: 2MHz (becomes 5)
- 0x0102: eighth NTSC (becomes 6)
- flags AND 0xcc: chipType (int)
- 0x00: Sega PSG (becomes 0)
- 0x04: TI SN76489 (becomes 1)

- 0x08: SN with Atari-like short noise (becomes 2)
- 0x0c: Game Gear (becomes 3)
- 0x40: TI SN76489A (becomes 4)
- 0x44: TI SN76496 (becomes 5)
- 0x48: NCR 8496 (becomes 6)
- 0x4c: Tandy PSSJ 3-voice sound (becomes 7)
- 0x80: TI SN94624 (becomes 8)
- 0x84: TI SN76494 (becomes 9)
- bit 4: noPhaseReset (bool)

0x04: Game Boy

- bits 0-1: chipType (int)
- 0: DMG (rev B)
- 1: CGB (rev C)
- 2: CGB (rev E)
- 3: AGB
- bit 3: noAntiClick (bool)

0x05: PC Engine

- bit 0: clockSel (int)
- 0: NTSC
- 1: pseudo-PAL
- bit 2: chipType (int)
- 0: HuC6280
- 1: HuC6280A
- bit 3: noAntiClick (bool)

0x06: NES, 0x88: VRC6, 0x8a: FDS and 0x8b: MMC5

- flags: clockSel (int)
- 0: NTSC (2A03)
- 1: PAL (2A07)
- 2: Dendy

0x07: C64 (8580) and 0x47: C64 (6581)

- bit 0-3: clockSel (int)
- 0: NTSC
- 1: PAL

- 2: SSI 2001

0x08: Arcade (YM2151+SegaPCM; COMPOUND)

- bit 0-7: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 4MHz
- this clock only applies to the YM2151.

0x09: Neo Geo CD (YM2610), 0xa5: Neo Geo (YM2610), 0xa6: Neo Geo extended (YM2610), 0x49: Neo Geo CD extended, 0x9e: YM2610B and 0xde: YM2610B extended

- bit 0-7: clockSel (int)
- 0: 8MHz
- 1: 8.06MHz (Neo Geo AES)

0x80: AY-3-8910

- bit 0-3: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: ZX Spectrum 48K (1.75MHz)
- 3: 2MHz
- 4: 1.5MHz
- 5: 1MHz
- 6: Sunsoft 5B
- 7: PAL NES
- 8: Sunsoft 5B on PAL NES
- 9: 1.10MHz
- 10: 2^21MHz
- 11: double NTSC
- 12: 3.6MHz
- 13: 1.25MHz
- 14: 1.536MHz
- bit 4-5: chipType (int)
- 0: AY-3-8910
- 1: YM2149(F)
- 2: Sunsoft 5B

- 3: AY-3-8914
- bit 6: stereo (bool)
- bit 7: halfClock (bool)
- bit 8-15: stereoSep (int)

0x81: Amiga

- bit 0: clockSel (int)
- 0: NTSC
- 1: PAL
- bit 1: chipType (int)
- 0: Amiga 500
- 1: Amiga 1200
- bit 2: bypassLimits (bool)
- bit 8-14: stereoSep (int)

0x82: YM2151 alone

- bit 0-7: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 4MHz

0x83: YM2612 alone, 0xa0: YM2612 extended, 0xbd: YM2612 extra features extended and 0xbe: YM2612 extra features

- bit 31: ladderEffect (bool)
- bit 0-30: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 8MHz
- 3: Firecore (`COLOR_NTSC*12/7`)
- 4: System 32 (`COLOR_NTSC*9/4`)

0x84: TIA

- bit 0: clockSel (int)
- 0: NTSC
- 1: PAL
- bit 1-2: mixingType (int)
- 0: mono

- 1: mono (no distortion)
- 2: stereo

0x85: VIC-20

- bit 0: clockSel (int)
- 0: NTSC
- 1: PAL

0x87: SNES

- bit 0-6: volScaleL (int)
- bit 8-14: volScaleR (int)

0x89: OPLL (YM2413) and 0xa7: OPLL drums (YM2413)

- bit 0-3: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 4MHz
- 3: half NTSC
- bit 4-31: patchSet (int)
- 0: YM2413
- 1: YMF281
- 2: YM2423
- 3: VRC7

0x8c: Namco 163

- bit 0-3: clockSel (int)
- 0: NTSC (2A03)
- 1: PAL (2A07)
- 2: Dendy
- bit 4-6: channels (int)
- bit 7: multiplex (bool)

0x8d: YM2203 and 0xb6: YM2203 extended

- bit 0-4: clockSel (int)
- 0: NTSC
- 1: PAL

- 2: 4MHz
- 3: 3MHz
- 4: 3.99MHz
- 5: 1.5MHz
- bit 5-6: prescale (int)
- 0: /6
- 1: /3
- 2: /2

0x8e: YM2608 and 0xb7: YM2608 extended

- bit 0-4: clockSel (int)
- 0: 8MHz
- 1: 7.98MHz
- bit 5-6: prescale (int)
- 0: /6
- 1: /3
- 2: /2

0x8f: OPL (YM3526), 0xa2: OPL drums (YM3526), 0x90: OPL2 (YM3812), 0xa3: OPL2 drums (YM3812), 0xb2: Yamaha Y8950 and 0xb3: Yamaha Y8950 drums

- bit 0-7: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 4MHz
- 3: 3MHz
- 4: 3.99MHz
- 5: 3.5MHz

0x91: OPL3 (YMF262) and 0xa4: OPL3 drums (YMF262)

- bit 0-7: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 14MHz
- 3: 16MHz
- 4: 15MHz

0x93: Intel 8253 (beeper)

- bit 0-1: speakerType (int)
- 0: unfiltered
- 1: cone
- 2: piezo
- 3: system

0x95: RF5C68

- bit 0-3: clockSel (int)
- 0: 8MHz
- 1: 10MHz
- 2: 12.5MHz
- bit 4-31: chipType (int)
- 0: RF5C68
- 1: RF5C164

0x97: Philips SAA1099

- flags: clockSel (int)
- 0: 8MHz
- 1: NTSC
- 2: PAL

0x98: OPZ (YM2414)

- flags: clockSel (int)
- 0: NTSC
- 1: pseudo-PAL
- 2: 4MHz

0x9a: AY8930

- bit 0-3: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: ZX Spectrum 48K (1.75MHz)
- 3: 2MHz
- 4: 1.5MHz
- 5: 1MHz
- 6: Sunsoft 5B

- 7: PAL NES
- 8: Sunsoft 5B on PAL NES
- 9: 1.10MHz
- 10: 2^21MHz
- 11: double NTSC
- 12: 3.6MHz
- bit 6: stereo (bool)
- bit 7: halfClock (bool)
- bit 8-15: stereoSep (int)

0x9d: VRC7

- bit 0-3: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 4MHz
- 3: half NTSC

0x9f: ZX Spectrum (beeper)

- bit 0-1: clockSel (int)
- 0: NTSC
- 1: PAL

0xa1: Konami SCC and 0xb4: Konami SCC+

- bit 0-6: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 1.5MHz
- 3: 2MHz

0xaa: MSM6295

- bit 0-6: clockSel (int)
- 0: 1MHz
- 1: 1.056MHz
- 2: 4MHz
- 3: 4.224MHz
- 4: NTSC
- 5: half NTSC
- 6: 2/7 NTSC
- 7: quarter NTSC

- 8: 2MHz
- 9: 2.112MHz
- 10: 875KHz
- 11: 937.5KHz
- 12: 1.5MHz
- 13: 3MHz
- 14: 1/3 NTSC
- bit 7: rateSel (bool)

0xab: MSM6258

- flags: clockSel (int)
- 0: 4MHz
- 1: 4.096MHz
- 2: 8MHz
- 3: 8.192MHz

0xae: OPL4 (YMF278B) and 0xaf: OPL4 drums (YMF278B)

- bit 0-7: clockSel (int)
- 0: NTSC
- 1: PAL
- 2: 33.8688MHz

0xb0: Seta/Allumer X1-010

- bit 0-3: clockSel (int)
- 0: 16MHz
- 1: 16.67MHz
- bit 4: stereo (bool)

0xb1: Ensoniq ES5506

- bit 0-4: channels (int)

0xb5: tildearrow Sound Unit

- bit 0: clockSel (int)
- 0: NTSC
- 1: PAL
- bit 2: echo (bool)

- bit 3: swapEcho (bool)
- bit 4: sampleMemSize (int)
- 0: 8K
- 1: 64K
- bit 5: pdm (bool)
- bit 8-13: echoDelay (int)
- bit 16-19: echoFeedback (int)
- bit 20-23: echoResolution (int)
- bit 24-31: echoVol (int)

0xb8: YMZ280B

- bit 0-7: clockSel (int)
- 0: 16.9344MHz
- 1: NTSC
- 2: PAL
- 3: 16MHz
- 4: 16.67MHz
- 5: 14MHz

0xc0: PCM DAC

- bit 0-15: rate (int)
- add +1 to this value
- bit 16-19: outDepth (int)
- bit 20: stereo (bool)

0xe0: QSound

- bit 0-11: echoDelay (int)
- bit 12-19: echoFeedback (int)

new Furnace instrument format

the main issue with Furnace instrument files is that they are too big, even if the instrument is nothing more than the FM setup...

the aim of this new format is to greatly reduce the size of a resulting instrument.

information

this format is "featural", meaning that only used parameters are stored (depending on instrument types).
this is the biggest improvement over the previous format, which stored everything including unused parameters.

features which are not recognized by Furnace will be ignored.

instruments are not compressed using zlib, unlike Furnace songs.

all numbers are little-endian.

the following fields may be found in "size":
- `f` indicates a floating point number.
- STR is a UTF-8 zero-terminated string.
- `???` is an array of variable size.
- S?? is an array of STRs.
- 1?? is an array of bytes.
- 2?? is an array of shorts.
- 4?? is an array of ints.

the format may change across versions. a `(>=VER)` indicates this field is only present starting from format version VER, and `(<VER)` indicates this field is present only before version VER.

furthermore, an `or reserved` indicates this field is always present, but is reserved when the version condition is not met.

the `size of this block` fields represent the size of a block excluding the ID and the aforementioned field.

header

.fui files use the following header:

```
size | description
-----|-----------------------------------
  4  | "FINS" format magic
  2  | format version
  2  | instrument type
 ??? | features...
```

instruments in a .fur file use the following header instead:

```
size | description
-----|-----------------------------------
  4  | "INS2" block ID
  4  | size of this block
  2  | format version
  2  | instrument type
 ??? | features...
```

a feature uses the following format:

```
size | description
-----|-----------------------------------
  2  | feature code
  2  | length of block
 ??? | data...
```

the following instrument types are available:

- 0: SN76489
- 1: FM (OPN)
- 2: Game Boy
- 3: C64
- 4: Amiga/sample
- 5: PC Engine
- 6: AY-3-8910
- 7: AY8930
- 8: TIA
- 9: SAA1099
- 10: VIC
- 11: PET
- 12: VRC6
- 13: OPLL
- 14: OPL
- 15: FDS
- 16: Virtual Boy
- 17: Namco 163
- 18: SCC
- 19: OPZ
- 20: POKEY
- 21: PC Speaker

- 22: WonderSwan
- 23: Lynx
- 24: VERA
- 25: X1-010
- 26: VRC6 (saw)
- 27: ES5506
- 28: MultiPCM
- 29: SNES
- 30: Sound Unit
- 31: Namco WSG
- 32: OPL (drums)
- 33: FM (OPM)
- 34: NES
- 35: MSM6258
- 36: MSM6295
- 37: ADPCM-A
- 38: ADPCM-B
- 39: SegaPCM
- 40: QSound
- 41: YMZ280B
- 42: RF5C68
- 43: MSM5232
- 44: T6W28
- 45: K007232
- 46: GA20
- 47: Pokémon Mini/QuadTone
- 48: SM8521
- 49: PV-1000

the following feature codes are recognized:

- NA: instrument name
- FM: FM ins data
- MA: macro data
- 64: C64 ins data
- GB: Game Boy ins data
- SM: sample ins data
- 01: operator 1 macros
- 02: operator 2 macros
- 03: operator 3 macros
- 04: operator 4 macros
- LD: OPL drums mode data
- SN: SNES ins data
- N1: Namco 163 ins data
- FD: FDS/Virtual Boy ins data

- WS: wavetable synth data
- SL: list of samples
- WL: list of wavetables
- MP: MultiPCM ins data
- SU: Sound Unit ins data
- ES: ES5506 ins data
- X1: X1-010 ins data
- EN: end of features
- if you find this feature code, stop reading the instrument.
- it will usually appear only when there sample/wave lists.
- instruments in a .fur shall end with this feature code.

instrument name (NA)

```
size | description
-----|-----------------------------------
 STR | instrument name
```

FM data (FM)

- FM operator order is:
- 1/3/2/4 (internal order) for OPN, OPM, OPZ and OPL 4-op
- 1/2/?/? (? = unused) for OPL 2-op and OPLL

```
size | description
-----|-----------------------------------
  1  | flags
     | - bit 4-7: op enabled
     |   - op order from 4 to 7: 0, 2, 1, 3
     |   - 2-op instruments: 0, 1, x, x
     | - bit 0-3: op count
-----|-----------------------------------
     | **base data**
     | /7 6 5 4 3 2 1 0|
  1  | |x| ALG |x| FB  |
  1  | |FMS2 |AMS| FMS |
  1  | |AM2|4| LLPatch |
-----|-----------------------------------
     | **operator data × opCount**
     | /7 6 5 4 3 2 1 0|
  1  | |r| D T |  MULT |
     |  \- KSR
  1  | |s|     T L     |
     |  \- SUS
  1  | |R S|v|   A R   |
     |      \- VIB
  1  | |A|KSL|   D R   |
     |  \- AM
  1  | |e|KVS|   D2R   |
```

```
     |  \- EGT
  1  | |  S L  |  R R  |
  1  | |  DVB  |  SSG  |
  1  | | DAM |DT2| W S |
```

macro data (MA)

notes:

- the macro range varies depending on the instrument type.
- "macro open" indicates whether the macro is collapsed or not in the instrument editor.
- meaning of extended macros varies depending on instrument type.
- meaning of panning macros varies depending on instrument type:
- for hard-panned chips (e.g. FM and Game Boy): left panning is 2-bit panning macro (left/right)
- otherwise both left and right panning macros are used

```
size | description
-----|-----------------------------------
  2  | length of macro header
 ??? | data...
```

each macro is represented like this:

```
size | description
-----|-----------------------------------
  1  | macro code
     | - 0: vol
     | - 1: arp
     | - 2: duty
     | - 3: wave
     | - 4: pitch
     | - 5: ex1
     | - 6: ex2
     | - 7: ex3
     | - 8: alg
     | - 9: fb
     | - 10: fms
     | - 11: ams
     | - 12: panL
     | - 13: panR
     | - 14: phaseReset
     | - 15: ex4
     | - 16: ex5
     | - 17: ex6
     | - 18: ex7
     | - 19: ex8
     | - 255: stop reading and move on
  1  | macro length
  1  | macro loop
  1  | macro release
```

```
  1  | macro mode
  1  | macro open/type/word size
     | - bit 6-7: word size
     |   - 0: 8-bit unsigned
     |   - 1: 8-bit signed
     |   - 2: 16-bit signed
     |   - 3: 32-bit signed
     | - bit 1-2: type
     |   - 0: normal
     |   - 1: ADSR
     |   - 2: LFO
     | - bit 0: open
  1  | macro delay
  1  | macro speed
 ??? | macro data
     | - length: macro length × word sizs
```

interpreting macro mode values

- sequence (normal): I think this is obvious...
- ADSR:
- `val[0]`: bottom
- `val[1]`: top
- `val[2]`: attack
- `val[3]`: hold time
- `val[4]`: decay
- `val[5]`: sustain level
- `val[6]`: sustain hold time
- `val[7]`: decay 2
- `val[8]`: release
- LFO:
- `val[11]`: speed
- `val[12]`: waveform
    - 0: triangle
    - 1: saw
    - 2: pulse
- `val[13]`: phase
- `val[14]`: loop
- `val[15]`: global (not sure how will I implement this)

C64 data (64)

```
size | description
-----|-----------------------------------
  1  | flags 1
     | - bit 7: dutyIsAbs
     | - bit 6: initFilter
     | - bit 5: volIsCutoff
```

```
     | - bit 4: toFilter
     | - bit 3: noise on
     | - bit 2: pulse on
     | - bit 1: saw on
     | - bit 0: triangle on
  1  | flags 2
     | - bit 7: oscSync
     | - bit 6: ringMod
     | - bit 5: noTest
     | - bit 4: filterIsAbs
     | - bit 3: ch3off
     | - bit 2: band pass
     | - bit 1: high pass
     | - bit 0: low pass
  1  | attack/decay
     | - bit 4-7: attack
     | - bit 0-3: decay
  1  | sustain release
     | - bit 4-7: sustain
     | - bit 0-3: release
  2  | duty
  2  | cutoff/resonance
     | - bit 12-15: resonance
     | - bit 0-10: cutoff
```

Game Boy data (GB)

```
size | description
-----|-----------------------------------
  1  | envelope params
     | - bit 5-7: length
     | - bit 4: direction
     | - bit 0-3: volume
  1  | sound length
     | - 64 is infinity
  1  | flags
     | - bit 1: always init envelope
     | - bit 0: software envelope (zombie mode)
  1  | hardware sequence length
 ??? | hardware sequence...
     | - length: 3*hwSeqLen
```

a value in the hardware sequence has the following format:

```
size | description
-----|-----------------------------------
  1  | command
     | - 0: set envelope
     | - 1: set sweep
     | - 2: wait
     | - 3: wait for release
     | - 4: loop
     | - 5: loop until release
  2  | data
```

```
    | - for set envelope:
    |   - 1 byte: parameter
    |     - bit 4-7: volume
    |     - bit 3: direction
    |     - bit 0-2: length
    |   - 1 byte: sound length
    | - for set sweep:
    |   - 1 byte: parameter
    |     - bit 4-6: length
    |     - bit 3: direction
    |     - bit 0-2: shift
    |   - 1 byte: nothing
    | - for wait:
    |   - 1 byte: length (in ticks)
    |   - 1 byte: nothing
    | - for wait for release:
    |   - 2 bytes: nothing
    | - for loop/loop until release:
    |   - 2 bytes: position
```

sample ins data (SM)

```
size | description
-----|-----------------------------------
  2  | initial sample
  1  | flags
     | - bit 2: use wave
     | - bit 1: use sample
     | - bit 0: use sample map
  1  | waveform length
 4?? | sample map... (120 entries)
     | - only read if sample map is enabled
```

the sample map format:

```
size | description
-----|-----------------------------------
  2  | note to play
  2  | sample to play
```

operator macro data (O1, O2, O3 and O4)

similar to macro data, but using these macro codes:

- 0: AM
- 1: AR
- 2: DR
- 3: MULT
- 4: RR
- 5: SL
- 6: TL

- 7: DT2
- 8: RS
- 9: DT
- 10: D2R
- 11: SSG-EG
- 12: DAM
- 13: DVB
- 14: EGT
- 15: KSL
- 16: SUS
- 17: VIB
- 18: WS
- 19: KSR

OPL drums mode data (LD)

```
size | description
-----|-----------------------------------
  1  | fixed frequency mode
  2  | kick freq
  2  | snare/hat freq
  2  | tom/top freq
```

SNES data (SN)

```
size | description
-----|-----------------------------------
  1  | attack/decay
     | - bit 4-6: decay
     | - bit 0-3: attack
  1  | sustain/release
     | - bit 5-7: sustain
     | - bit 0-4: release
  1  | flags
     | - bit 4: envelope on
     | - bit 3: make sustain effective (<131)
     | - bit 0-2: gain mode
     |   - 0: direct
     |   - 4: dec
     |   - 5: exp
     |   - 6: inc
     |   - 7: bent
  1  | gain
  1  | decay 2/sustain mode (>=131)
     | - bit 5-6: sustain mode
     |   - 0: direct
     |   - 1: sustain (release with dec)
     |   - 2: sustain (release with exp)
     |   - 3: sustain (release with rel)
     | - bit 0-4: decay 2
```

Namco 163 data (N1)

```
size | description
-----|-----------------------------------
  4  | waveform
  1  | wave pos
  1  | wave len
  1  | wave mode
```

FDS/Virtual Boy data (FD)

```
size | description
-----|-----------------------------------
  4  | mod speed
  4  | mod depth
  1  | init mod table with first wave
 1?? | modulation table (32 entries)
```

wavetable synth data (WS)

```
size | description
-----|-----------------------------------
  4  | first wave
  4  | second wave
  1  | rate divider
  1  | effect
     | - bit 7: single or dual effect
  1  | enabled
  1  | global
  1  | speed (+1)
  1  | parameter 1
  1  | parameter 2
  1  | parameter 3
  1  | parameter 4
```

list of samples (SL)

```
size | description
-----|-----------------------------------
  1  | number of samples
 1?? | sample indexes...
 4?? | pointers to samples...
     | - these use the Furnace sample format.
```

list of wavetables (WL)

```
size | description
-----|-----------------------------------
  1  | number of wavetables
```

```
 1?? | wavetable indexes...
 4?? | pointers to wavetables...
     | - these use the Furnace wavetable format.
```

MultiPCM data (MP)

```
size | description
-----|-----------------------------------
  1  | attack rate
  1  | decay 1 rate
  1  | decay level
  1  | decay 2 rate
  1  | release rate
  1  | rate correction
  1  | LFO rate
  1  | vibrato depth
  1  | AM depth
```

Sound Unit data (SU)

```
size | description
-----|-----------------------------------
  1  | switch roles of phase reset timer and frequency
```

ES5506 data (ES)

```
size | description
-----|-----------------------------------
  1  | filter mode
     | - 0: HPK2_HPK2
     | - 1: HPK2_LPK1
     | - 2: LPK2_LPK2
     | - 3: LPK2_LPK1
  2  | K1
  2  | K2
  2  | envelope count
  1  | left volume ramp
  1  | right volume ramp
  1  | K1 ramp
  1  | K2 ramp
  1  | K1 slow
  1  | K2 slow
```

X1-010 data (X1)

```
size | description
-----|-----------------------------------
  4  | bank slot
```

ZSM format specification

Zsound Repo

ZSM is part of the Zsound suite of Commander X16 audio tools found at:

https://github.com/ZeroByteOrg/zsound/

Current ZSM Revision: 1

ZSM is a standard specifying both a data stream format and a file structure for containing the data stream. This document provides the standard for both the ZSM stream format and for the ZSM container file format.

Whenever it becomes necessary to modify the ZSM standard in such a way that existing software will not be compatible with files using the newer standard, this version number will be incremented, up to a maximum value of 254.

Version 255 (-1) is reserved for internal use by the player

Headerless Data File Format:

Since Kernal version r39, it is possible to load data files that do not have the CBM 2-byte load-to-address header. As of version r41, this functionality is equally accessible in the standard interactive BASIC interface. As the "PRG" header is no longer necessary, ZSM files will NOT contain this header in order to appear as any other common data file such as `.wav`, `.png`, etc. As such, users and programs must use the "headerless mode" when loading a ZSM into memory on the Commander X16. The previously-suggested dummy PRG header has been incorporated to the ZSM header as a magic header for file identity verification purposes.

ZSM file composition

OFFSET	LENGTH	FIELD
0x00	16	ZSM HEADER
0x10	variable	ZSM STREAM
?	?	(optional) PCM HEADER
?	variable	(optional) PCM DATA

ZSM Header

The ZSM header is 16 bytes long.

- All multi-byte values are little endian unless specified otherwise
- All offsets are relative to the beginning of the ZSM header

OFFSET	LENGTH	FIELD	DESCRIPTION
0x00	2	Magic Header	The string 'zm' (binary 0x7a 0x6d)
0x02	1	Version	ZSM Version. 0-0xFE (0xFF is reserved)
0x03	3	Loop Point	Offset to the starting point of song loop. 0 = no loop.
0x06	3	PCM offset	Offset to the beginning of the PCM index table (if present). 0 = no PCM header or data is present.
0x09	1	FM channel mask	Bit 0-7 are set if the corresponding OPM channel is used by the music.
0x0a	2	PSG channel mask	Bits 0-15 are set if the corresponding PSG channel is used by the music.
0x0c	2	Tick Rate	The rate (in Hz) for song delay ticks. *60Hz (one tick per frame) is recommended.*
0x0e	2	reserved	Reserved for future use. Set to zero.

ZSM Music Data Stream Format

BYTE 0	BYTE 1 (VARIABLE)	BYTE N	BYTE N+1 (VARIABLE)	...	END OF STREAM
CMD	DATA	CMD	DATA	...	0x80

CMD (command) byte values

CMD bytes are bit-packed to hold a command Type ID and a value (n) as follows:

CMD	BIT PATTERN	TYPE	ARG. BYTES	ACTION
0x00-0x3F	`00`nnnnnn	PSG write	1	Write the following byte into PSG register offset *n*. (from 0x1F9C0 in VRAM)
0x40	`01000000`	EXTCMD	1+?	

CMD	BIT PATTERN	TYPE	ARG. BYTES	ACTION
				The following byte is an extension command. (see below for EXTCMD syntax)
0x41-0x7F	01nnnnnn	FM write	2*n*	Write the following *n* reg/val pairs into the YM2151.
0x80	10000000	EOF	0	This byte MUST be present at the end of the data stream. Player may loop or halt as necessary.
0x81-0xFF	1nnnnnnn	Delay	0	Delay *n* ticks.

EXTCMD:

The EXTCMD byte is formatted as ccnnnnnn where c=channel and n=number of bytes that follow. If the player wishes to ignore a channel, it can simply advance n bytes and continue processing. See EXTCMD Channel Specifications below for more details.

PCM Header

The size and contents of the PCM header table is not yet decided. This will depend largely on the strucure of EXTCMD channel 0, and be covered in detail in that specification.

Any offset values contained in the PCM data header block will be relative to the beginning of the PCM header, not the ZSM header. The intention is to present the digital audio portion as a set of digi clips ("samples" in tracker terminology) whose playback can be triggered by EXTCMD channel zero.

PCM Sample Data

This will be a blob of PCM data with no internal formatting. Indeces / format information / loop points / etc regarding this blob will be provided via the PCM header. The end of this blob will be the end of the ZSM file.

EXTCMD Channel Scifications

Extension commands provide optional functionality within a ZSM music file. EXTCMD may be ignored by any player. EXTCMD defines 4 "channels" of message streams. Players may implement support for any, all, or none

of the channels as desired. An EXTCMD may specify up to 63 bytes of data. If more data than this is required, then it must be broken up into multiple EXTCMDs.

EXTCMD in ZSM stream context:

...	CMD 0X40	EXTCMD	N BYTES	CMD	...

EXTCMD byte format:

BIT PATTERN	C	N
ccnnnnnn	Extension Channel ID	Number of bytes that follow

EXTCMD Channels:

1. PCM instrument channel
2. Expansion Sound Devices
3. Synchronization events
4. Custom

The formatting of the data within these 4 channels is presently a work in progress. Definitions for channels 0-3 will be part of the official ZSM specifications and implemented in the Zsound library. Significant changes within one of these three channels' structure may result in a new ZSM version number being issued. The formatting and content of the 3 official EXTCMD channels will be covered here.

The Custom channel data may take whatever format is desired for any particular purpose with the understanding that the general ecosystem of ZSM-aware applications will most likely ignore them.

EXTCMD Channel:

0: PCM audio

The structure of data within this channel is not yet defined.

1: Expansion Sound Devices

This channel is for data intended for "well-known" expansion hardware used with the Commander X16. As the community adopts various expansion hardware, such devices will be given a standard "ID" number so that

all ZSM files will agree on which device is being referenced by expansion HW data.

The specification of new chip IDs should not affect the format of ZSM itself, and thus will not result in a ZSM version update. Players will simply need to update their list of known hardware.

Players implementing this channel should implement detection routines during init to determine which (if any) expansion hardware is present. Any messages intended for a chip that is not present in the system should be skipped.

An expansion HW write will contain the following data:

CHIP ID	NUBER OF WRITES (N)	N TUPLES OF DATA
one byte	one byte	N * tuple_size bytes

- The total number of bytes MUST equal exactly the number of bytes specified in the preceding EXTCMD.
- The tuple_size is determined by the needs of the device, and thus will be specified per-device along with its chip ID assignment. This is likely to be 1-3 bytes for most devices.

There are currently no supported expansion HW IDs assigned.

2: Synchronization Events

The purpose of this channel is to provide for music synchronization cues that applications may use to perform operations in sync with the music (such as when the Goombas jump in New Super Mario Bros in time with the BOP! BOP! notes in the music). It is intended for the reference player to provide a sync channel callback, passing the data bytes to the callback function, and then to proceed with playback.

The data structure within this channel is not yet defined. It is our intention to work with the community in order to collaborate on a useful structure.

3: Custom

The purpose for this channel is that any project with an idea that does not fit neatly into the above categories may pack data into the project's music files in whatever form is required. It should be understood that these ZSMs will not be expected to use the extended behaviors outside of the project they were designed for. The music itself, however, should play properly.

The only constraint is that the data must conform to the EXTCMD byte - supplying exactly the specified number of bytes per EXTCMD.

The reference playback library in Zsound will implement this channel as a simple callback passing the memory location and data size to the referenced function, and take no further action internally.