FluidSynth
Adding poly/mono functionality

jean-jacques ceresa

Patch 0001
- first writing 10/052015 PatchFluidPolyMono-0001.
- first coding PatchFluidPolyMono-0001 may-june 2016.

Patch 0002
- Correction typos errors 1/07/2016.
- Correction to get uniform compilation on Rpi2. Thanks to Ben Gonzales for reporting and testing on Rpi2.
- Correction in the method to apply the patch. Thanks to R.L Horn.

Patch 0003
21/07/2016
- Many enhancements functionality (see 1).
- Tutorials examples with commands files (see 2.1 and 3.1)
- Thanks to Ben Gonzales for informations about monophonic accoustic instruments and electronic winds instruments behaviors.

1. Introduction 4
1.1. PATCH ARCHIVE CONTENT 5
1.1.1. fluid-polymono-0003.patch content 5
1.1.2. How the patch has be done ? 5
*1.1.3. How to apply the patch using tree **./fluid-polymono-0003** ? 6*
*1.1.4. How to apply the patch using diff result: **fluid-polymono-0003.patch** ? 6*
2. Part1: Specifications Omni On/Off, Poly/Mono in FluidSynth 6
2.1. PART 1: TUTORIAL EXAMPLES – UNDERSTANDING POLY/MONO MODE 6
2.1.1. What are basic channels ? 6
2.1.2. What are default 'basic channels' in FluidSynth ? 6
2.1.3. How to change the whole set of actual basic channels ? 6
2.1.4. How to add a new basic channel among others actual basic channels ? 7
2.1.5. How to change an actual basic channel ? 7
2.1.6. How to change an existing basic channel and add a new one ? 8
2.1.7. How to know the state of one or more MIDI channels ? 8
2.1.8. Is there a way to set basic channels vial MIDI ? 9
2.1.9. Is there a way to change the mode of an actual basic channel via MIDI ? 9
2.2. "BASIC CHANNEL" NUMBER IN FLUIDSYNTH 9
2.3. MIDI MODES MESSAGES IN FLUIDSYNTH 9
2.3.1. Specification MIDI: Omni On/Off et Poly/Mono in FluidSynth 9
2.4. RECEIVER WITH ONE "BASIC CHANNEL" 9
*2.4.1. Listening control: **Omni On**, **Omni Off** 9*
*2.4.2. Mode polyphonic or monophonic: **Poly On**, **Mono On**. 9*
2.5. RECEIVER WITH MORE THAN ONE "BASIC CHANNEL" 11
2.5.1. Using Omni On (mode 1 and 2) with more than one "Basic Channel" 11
2.6. POLY/MONO MODE API IN FLUIDSYNTH 12
*2.6.1. Reset basic channels: **fluid_synth_reset_basic_channels(n, BasicChannelsInfos)** 12*
*2.6.2. Get Basic Channels count: **fluid_synth_count_basic_channels()** 12*

*2.6.3. Get basic channels: **fluid_synth_get_basic_channels()*** *12*
*2.6.4. Set basic channel: **fluid_synth_set_basic_channel(chan,mode, val)*** *13*
*2.6.5. Get channel mode: **fluid_synth_get_channel_mode(chan, modeInfo)*** *13*
2.7. POLY/MONO MODES COMMANDS IN FLUIDSYNTH 14
*2.7.1. Command to get MIDI basic channels number: **basicchannels*** *14*
*2.7.2. Command to replace MIDI basic channels: **resetbasicchannels*** *14*
*2.7.3. command to change/add MIDI basic channels : **setbasicchannels*** *14*
*2.7.4. Command to read MIDI channels mode: **channelsmode*** *15*
2.8. PART 1: IMPLEMENTATIONS STEPS IN FLUIDSYNTH. 15
2.8.1. Implementation steps: Poly/mono mode API (2.6) *15*
2.8.2. Implementation steps: Poly/mono mode commands (2.7) *17*
2.8.3. Inplementation steps: MIDI mode messages (2.3) *17*
2.8.4. ignoring MIDI messages on dsabled MIDI channels *18*
2.8.5. MIDI CC MIDI global *19*
3. Part 2:Polyphonic/monophonic behavior inside Fluidsynth 19
3.1. PART 2: TUTORIAL EXAMPLES- UNDERSTANDING MONOPHONIC BEHAVIOR 19
3.1.1. How to set a MIDI channel in mono mode ? *19*
3.1.2. Why using legato pedal On/Off ? *20*
3.1.3. How reacts FluidSynth when the musician plays legato or staccato ? *20*
3.1.4. What are legato mode ? *20*
3.1.5. What are breath mode ? *21*
3.1.6. What is portamento ? *21*
3.1.7. What are portamento mode ? *21*
3.1.8. What about sustained note in monophonic mode ? *22*
3.2. MONOPHONIC MIDI WIND CONTROLLER INSTRUMENT 22
3.2.1. Basic behavior *22*
3.2.2. Playing staccato *22*
3.2.3. Playing legato: n1,n2,n1 ..... *23*
3.3. POLYPHONIC CONTROLLER (KEYBOARD) 24
3.3.1. basic behavior *24*
3.3.2. Playing staccato *24*
3.3.3. Playing legato *24*
3.4. SYNTHESIZER MONOPHONIC BEHAVIOR 24
3.4.1. Input controller (mono/poly) *24*
3.4.2. polyphonic controller (keyboard) to monophonic channel. *24*
3.4.3. Legato playing detector – the monophonic list *25*
3.4.4. CC Breath controller to monophonic channel *25*
3.4.5. How FluidSynth can play CC Breath controller *25*
3.4.6. Monophonic controller (MIDI Wind Controller) to monophonic channel. *26*
3.4.7. Using Sustain / Sostenuto in monophonic mode *27*
3.4.8. Useful MIDI CC in monophonic mode *27*
3.4.9. Portamento On/On *27*
3.4.10. Legato On/On *27*
3.4.11. CC Portamento Control (PTC) *27*
3.4.12. CC Global to control all channels at the same time (Mode 3) (OmniOff- Mono). *28*
3.5. POLYPHONIC CHANNEL BEHAVIOR 28
3.6. MONOPHONIC LIST IMPLEMENTATION 28
3.6.1. List initialisation *28*
3.7. LEGATO MODES 28
3.7.1. Mode 0: "retrigger_0" (fast release) *29*
3.7.2. Mode 1: "retrigger_1" (normal release) *29*
3.7.3. Mode 2: "multi-retrigger" *30*
3.7.4. Mode 3: "single-trigger_0" *31*
3.7.5. Mode 4: "single-trigger_1" *31*
3.7.6. Legato playing through InstrumentZone and PresetZone in SoundFont *31*
*3.7.7. API set legato mode: **fluid_synth_set_legato_mode(chan,mode)*** *32*
*3.7.8. API get legato mode : **fluid_synth_get_legato_mode(chan,mode)*** *32*
*3.7.9. command to set legato mode:**setlegatomode*** *32*
*3.7.10. command to print legato mode:**legatomode*** *33*
3.8. BREATH MODE 33
*3.8.1. API get defaulth breath mode : **fluid_synth_set_breath_mode(chan, breathmode)*** *33*
*3.8.2. API get breath mode : **fluid_synth_get_ breath_mode(chan,breathmode)*** *33*
*3.8.3. command to set breath mode:**setbreathmode*** *34*

*3.8.4. command to print breath mode:**breathmode*** *34*
3.9. PORTAMENTO MODE 34
3.9.1. Portamento 'legato only' in mono mode *34*
3.9.2. Portamento 'staccato only' or 'each note' in mono mode *35*
3.9.3. Portamento 'legato only' in poly mode *35*
3.9.4. Portamento 'staccato' in poly mode: *35*
*3.9.5. API set portamento mode: **fluid_synth_set_portamento_mode(chan,mode)*** *35*
*3.9.6. API get portamento mode : **fluid_synth_get_portamento_mode(chan,mode)*** *35*
*3.9.7. command to set portamento mode:**setportamentomode*** *36*
*3.9.8. command to print portamento mode:**portamentomode*** *36*
3.10. MONOPHONIC MODE IMPLEMENTATION IN FLUIDSYNTH 36
3.10.1. ignore MIDI message on MIDI channel disabled *36*
3.10.2. Insertion point of Poly/mono mode on noteOn et note Off *37*
3.10.3. fluid_synth_noteon_LOCAL() *37*
3.10.4. fluid_synth_noteoff_LOCAL() *38*
3.10.5. Using fluidsynth router to simulate a legato pedal by Sustain pedal *38*
3.10.6. monophonic algorithm implementation *38*
3.11. PART 2: IMPLEMENTATIONS STEPS IN FS. 38
3.11.1. integrate Polyphonic/monophonic on noteOn and note Off *38*
3.11.2. Adding monophonic list *39*
3.11.3. Monophonic algorithm *40*
*3.11.4. staccato noteOn: **fluid_synth_noteon_mono()*** *40*
*3.11.5. noteOff poly or mono: **fluid_synth_noteoff_monopoly ()*** *41*
*3.11.6. noteon mono legato: **fluid_synth_noteon_mono_legato ()*** *41*
*3.11.7. noteon mono legato: **fluid_synth_noteon_mono_legato (single_trigger())*** *42*
3.11.8. Portamento *42*
3.11.9. implementation API legato mode *43*
3.11.10. Implementation commands legato mode *44*
3.11.11. implementation API: mode Default Breath controller *44*
3.11.12. implementation: Commands Breath mode *45*
3.11.13. implementation: mode Default Breath controller *45*
3.11.14. sleep command *46*
3.11.15. implementation: legato trough more zones IZ and PZ *46*
3.11.16. Implementation of portamento request *49*
3.11.17. API portamento mode *50*
3.11.18. commands portamento mode *51*
3.11.19. Implementation Breath Sync noteOn/noteOff *51*
3.11.20. Using fluidsynth router to simulate a Breath controller using volume pedal *53*

1. Introduction

This document describes the adding Poly/mono functionality to FluidSynth library.
This patch supersedes previous poly-mono patch.
New fonctionnality supported by patch-0003 are marked in **bold.**
For clarity the document is split in 2 parts

Patch content (see 1.1), describes
- how the patch has be done, and
- how it can be applied to actual sources tree (v 1.1.6 git).

Functionalities are split in two part.
Part 1 (see 2)
This part describes Poly/Mono mode
- Tutorial examples (2.1). Useful examples to learn what is poly/mono mode
  Example files are directly executable on the console using the *source* command.
- This patch handle the MIDI specifications concept of 'basic channel' (see 2.2,2.4, 2.5).
- MIDI modes messages in Fluidsynth:Omni On/Off, Poly/Mono (see 2.3).
- New API for channel basic and Poly/Mono mode change(see 2.6).
- New shell commands for basic channel and Poly/Mono mode change(see 2.7).
- The Part 1 table that helps to understand the patch contents for those who want to see the details inside the code(see 2.8).This table enumerates the steps in the order of construction.

Part 2 (see 3)
This part describes mainly the monophonic behavior
- Tutorial examples (3.1).
  useful examples to learn what is mono mode behavior.
  Example files are directly executable on the console using the *source* command.
- Type of MIDI input controller (monophonic/polyphonic) (3.2, 3.3).
- MIDI CC messages handled (3.4.8):
  CC portamento(65d) On/Off (3.4.9)
  CC legato(68d) On/Off (3.4.10)
  CC portamento time (msb,lsb) (5, 37d)
  CC Portamento Control(84d) supported in Poly and Mono mode (3.4.11)**.**
  CC Global to control all channels at the same time (Mode 3) (OmniOff- Mono)(3.4.12).
- **Portamento mode (3.9)**
  New API, shell commands to handle portamento mode (3.9.5, 3.9.6, 3.9.7, 3.9.8).

- Use of sustain/sostenuto pedal in monophonic mode (3.4.7).
- Use of CC Breath(3.4.5).
  New API (3.8.1), shell commands to handle default breath to Attenuation modulator (3.8.3).
  New option **Breath Sync** to trigger noteOn/noteOff with the breath controller (3.8.3)
- Legato mode playing (3.7).
  New legato mode: Retrigger_1 (normal release) (3.7.2)
  New API (3.7.7) , shell commands to handle legato mode (3.7.9).
- The Part 2 table that helps to understand the patch contents contents for those who want to study the details inside the code (see 3.11).This table enumerates the steps in the order of construction.

No more limitation (see 3.7.6)
This patch now accept legato passage through multiples Instruments Zones and Preset Zone.This chapter describes the enhancement to suppress the know limitations of previous patches.

Things not handled
- Sysex message to handle channel basic.

1.1. Patch archive content

- Documentation: PatchFluidPolyMono-0003.pdf.
- Compete tree of file patched: ./fluid-polymono-0003
- Patch: fluid-polymono-0003.patch
- Soundfont for tutorials: Legato_demo.sf2
- Tutorials commands files examples:
  Legato mode: leg_0.txt, leg_1.txt, leg_2.txt , leg_3.txt, leg_4.txt
  Portamento: leg_por_0.txt, leg_por_1.txt, leg_por_2.txt , leg_por_3.txt, leg_por_4.txt

1.1.1. fluid-polymono-0003.patch content

This is the list of the file impacted by this patch

Base directory: fluisynth-1.1.6	**action on this file**
include/fluidsynth/types.h include/fluisynth/synth.h	patch adding poly mono
src/sfloader/fluid_defsfont.h src/sfloader/fluid_defsfont.c src/sfloader/fluid_ramsfont.c	patch adding poly mono
src/synth/fluid_chan.c src/synth/fluid_chan.h	patch adding poly mono
src/synth/fluid_voice.c src/synth/fluid_ voice.h	patch adding poly mono
src/synth/fluid_synth.c src/synth/fluid_synth.h	patch adding poly mono
src/synth/fluid_synth_polymono.c	new file
src/synth/fluid_synth_mono.c	new file
src/midi/fluid_midi.h	bug
src/bindings/fluid_cmd.c src/bindings/fluid_cmd.h	patch adding poly mono
src/rvoice/fluid_rvoice_event.c	patch adding poly mono
src/rvoice/fluid_rvoice.c src/rvoice/fluid_ rvoice.h	patch adding poly mono
src/rvoice/fluid_adsr_env.h	minor bug
src/CMakeLists.txt	file adding (fluid_synth_polymono.c, fluid_synth_mono.c)
src/Makefile.am	file adding (fluid_synth_polymono.c, fluid_synth_mono.c)

1.1.2. How the patch has be done ?

The patch has be done starting from the actual source tree coming from **git source repository (./fluidsynth-1.1.6)**
For convenience the full tree of sources already patched is given in the PatchFluidPolyMono-0003.zip **(./fluid-polymono-0003**). So this tree is directly usable to build on the target platform.

For conveniance a diff is given to get a full insight of the changes
diff -Naur ./fluidsynth-1.1.6 ./fluid-polymono-0003 > fluid-polymono-0003.patch
All changes are listed in chapters 2.8, 3.11.

Now you have 2 methods to build this patch on a target platform:
- The direct method using the provided tree already patched. (1.1.3).
- The indirect method building your own tree to be patched (1.1.4).

1.1.3. How to apply the patch using tree **./fluid-polymono-0003** ?

- Copy the provided directory tree **./fluid-polymono-0003** to the platform
- Then use the usual tools to build from this tree (cmake,make).

1.1.4. How to apply the patch using diff result: **fluid-polymono-0003.patch** ?

- get **./fluidsynth-1.1.6** the base directory source coming from **git source repository**

- Put **fluid-polymono-0003.patch** in **./fluidsynth-1.1.6** directory
- From this directory execute patch command.
  cd fluidsynth-1.1.6
  patch -p2 < fluid-polymono-0003.patch
- Now tree ./fluidsynth-1.1.6 has been patched,you may use the usual tools to build from this tree (cmake,make).

2. Part1: Specifications Omni On/Off, Poly/Mono in FluidSynth

Note this chapter is based on MIDI standard specification knowledge. It doesn't replace the MIDI specification. So report to this document when necessary: MIDI_MMA_Specification.pdf 1.0 version 4.2 1995.

2.1. Part 1: Tutorial examples – understanding poly/mono mode

This chapter describes useful examples to learn and understand what is poly/mono mode
Example files are directly executable on the console using the source command.

>source command-example-file

2.1.1. What are basic channels ?

Basic channels is a MIDI specification. It allows to split the whole set of MIDI channels in independants groups of MIDI channels. Each group can bet set in differents mode (poly, mono, omni on, omnioff). The first MIDI channel of a group is called 'basic channel'.

2.1.2. What are default 'basic channels' in FluidSynth ?

At initialization the default basic channel is defined by settings (see 2.2).

type the command ***basicchannels*** to display actual basic channels.
> **basicchannels**
Basic channel: 0, poly omni on (0), nbr: 16

Note: command file execution :**source poly_mono_0.txt**

2.1.3. How to change the whole set of actual basic channels ?

Assuming you want to set 2 groups of channels
Group 1: first channel 5 in poly mode, only one channel
Group 2: first channel 10 in mono mode , only one channel

Group 1 should have following settings:
- basic channel **5**, mode poly, omni off, (mode **2**).

Group 2 should have the following settings:
- basic channel **10**, mode mono, omni off, (mode 3), composed of **one** channel.

Use command ***resetbasicchannels***:
>**resetbasicchannels 5 2** 0 **10 3 1**

Use ***basicchannels*** command to verify your settings
> **basicchannels**
Basic channel: 5, poly omni off(2), nbr: 1
Basic channel: 10, mono omni off(3), nbr: 1

Note: command file execution :**source poly_mono_1.txt**

2.1.4. How to add a new basic channel among others actual basic channels ?

Perhaps you have already set several groups of basics channels and you want add a new one without modifying actual groups.
Assuming following actual groups:

- Basic channel: 5, poly omni off(mode 2), nbr: 1
- Basic channel: 10, mono omni off(mode 3), nbr: 1

Now we want to add a new group 3:
Group 3 should have the following settings:
- basic channel **13**, mode mono, omni off, (mode **3**) composed of **2** channels.

Use command ***setbasicchannels***:
>**setbasicchannels 13 3 2**

Use ***basicchannels*** command to verify your settings
> **basicchannels**
Basic channel: 5, poly omni off(2), nbr: 1
Basic channel: 10, mono omni off(3), nbr: 1
Basic channel: 13, mono omni off(3), nbr: 2
>

Note: command file execution :**source poly_mono_2.txt**

2.1.5. How to change an actual basic channel ?

Perhaps you have already set several groups of basics channels and you want change the settings of one.

Assuming following actual groups:
- Group 1:Basic channel: 5, poly omni off(mode 2), nbr: 1
- Group 2:Basic channel: 10, mono omni off(mode 3), nbr: 1
- Group 3:Basic channel: 13, mono omni off(mode 3), nbr: 2

Now we want to change group 1:
Group 1 should have the following settings:
- basic channel **5**, mode poly, omni on, (mode **0**) composed of **all** channels in this group.

Use command ***setbasicchannels***:
>**setbasicchannels 5 0** 0

Use ***basicchannels*** command to verify your settings
> **basicchannels**
Basic channel: 5, poly omni on (0), nbr: 5
Basic channel: 10, mono omni off(3), nbr: 1
Basic channel: 13, mono omni off(3), nbr: 2
>

Note: command file execution :**source poly_mono_3.txt**

2.1.6. How to change an existing basic channel and add a new one ?

Note that command **setbasicchannels** allows to add or change groups of basics channels.
Note also that the commands allows this for more than one groups executing only one command:

The following command restore Group 1 to the following state:

- Group 1:Basic channel: **5**, poly omni off(mode **2**), nbr: 1

Than add a new group:

- Group 0:Basic channel: **2**, mono omni on(mode **1**), composed of **all** channels in this group

Use command ***setbasicchannels*** to change a group and add a new one
>**setbasicchannels 5 2** 0 **2 1** 0

Use ***basicchannels*** command to verify your settings
> **basicchannels**
Basic channel: 2, mono omni on (1), nbr: 3
Basic channel: 5, poly omni off(2), nbr: 1
Basic channel: 10, mono omni off(3), nbr: 1
Basic channel: 13, mono omni off(3), nbr: 2
>

Note: command file execution :**source poly_mono_4.txt**

2.1.7. How to know the state of one or more MIDI channels ?

Use the command **channelsmode** [chan1 chan2 ….]

To display the state off alls MIDI channels
> **channelsmode**

Channel	Status	Type	Mode	Nbr of channels
channel: 0	disabled			
channel: 1	disabled			
channel: 2	enabled	basic channel	mono omni on (1)	nbr: 3
channel: 3	enabled	--	mono	--
channel: 4	enabled	--	mono	--
channel: 5	enabled	basic channel	poly omni off(2)	nbr: 1
channel: 6	disabled			
channel: 7	disabled			
channel: 8	disabled			
channel: 9	disabled			
channel: 10	enabled	basic channel,	mono omni off(3)	nbr: 1
channel: 11	disabled			
channel: 12	disabled			
channel: 13	enabled	basic channel	mono omni off(3)	nbr: 2
channel: 14	enabled	--	mono	--
channel: 15	disabled			

>

To display the state of MIDI channels 2,5, 10, 13 only
> **channelsmode 2 5 10 13**

Channel	Status	Type	Mode	Nbr of channels
channel: 2	enabled	basic channel	mono omni on (1)	nbr: 3
channel: 5	enabled	basic channel	poly omni off(2)	nbr: 1
channel: 10	enabled	basic channel	mono omni off(3)	nbr: 1
channel: 13	enabled	basic channel	mono omni off(3)	nbr: 2

>

Note: command file execution :**source poly_mono_5.txt**

2.1.8. Is there a way to set basic channels vial MIDI ?

No, actually in Fluidsynth there no way.

The MIDI specifications propose to use MIDI sysex messages to do that. To implement MIDI sysex messages manufacturers need to grant a unique ID from MMA or JMSC. This is not a free operation.

2.1.9. Is there a way to change the mode of an actual basic channel via MIDI ?

Yes, simply by sending MIDI CC poly On, mono On, omni On, Omni Off message on this basic channel. (see 2.3 for details)

2.2. "Basic Channel" number in FluidSynth

A fluisynth instance may have more than one Basic Channel.
So FluidSynth can work in more than one mode at the same time(see 2.5).
Inside FluidSynth Mode numbers are zero based: 0, 1, 2, 3.

- New settings **synth.basic-channel, synth.basic-channel.mode, synth.basic-channel-modeval**. allow to set only one basic channel.
  - setting **synth.basic-channel** (int type) is the basic channel number (defaut: 0).
  - setting **synth.basic-channel-mode** (int type) is the mode of this basic channel (defaut: 0, i.e Omni On Poly On).
  - setting **synth.basic-channel-modeval** (int type) is the number of MIDI channels (int type) for mode 3 (défaut 0).
- At creation time (new_fluid_synth()) the settings have default value (basic channel 0, mode 0,modeval 0).
  So by default fluidsynth is a polyphonic synthesizer on all MIDI channels.
- An application can use the API **fluid_synth_reset_basic_channels()**(2.6.1)**,** **fluid_synth_get_basic_channels()** (2.6.3) to change or read "Basic channels" informations.

- Default commands shell have new commands to set/print "basic channels informations". So a Fluidsynth instance can work in "multi mode" (see 2.7).

2.3. MIDI Modes messages in FluidSynth

2.3.1. Specification MIDI: Omni On/Off et Poly/Mono in FluidSynth

Following MIDI CC are handled :
Omni On, Omni Off, Poly On, Mono On only on Basic channel, otherwise messages are ignored.

2.4. Receiver with one "Basic Channel"

MIDI standard specify 2 CC messages **Omni On** and **Omni Off** to define if channels listening is global (**Omni On**) or limited to a channels range (**Omni Off**).
Basic Channel are to be set inside the receiver. (by sysex or others methods (see 2.7)).

2.4.1. Listening control: **Omni On**, **Omni Off**

Omni On : CC 125 Data=0
Allows listening on all MIDI channels MIDI (0 to 15).
Data field is 0.

Omni Off: CC 124 Data 0
Allows listening on a range relative to "**Basic Channel**" channel.
Data field is 0.

Remark: This message gives no information about the range(see Mono On 2.4.2).

2.4.2. Mode polyphonic or monophonic: **Poly On**, **Mono On**.

MIDI standard specify 2 other CC messages to set channels in polyphonic or monophonic mode.

Poly On: CC 127 Data=0
Data field is 0.
Set the MIDI channels in polyphonic.

- If Omni is On, all channels (0 to 16) are listened and polyphonic.
- If Omni is Off, as data field is 0, there is only one channel set in polyphonic (i.e "Basic channel"). Others channels aren't listened.

Mono On: CC 127 Data=M
Set the MIDI M channels in monophonic.

- If Omni is On, data field is ignored, all channels (0 à 16) are listened and monophonic.
- If Omni is Off, data field M is the MIDI channels range (relative to "Basic Channel "), listened in monophonic. Others channels aren't listened.
  Value M to 0 means all channels from "Basic Channel" .

Remark: MIDI standard specify to send Poly On, Mono On messages on "Basic Channel" number in order to be accepted.

MIDI standard allows the following combinations:

- Mode 1: Omni On , Poly On Data=0: All channels are listened in polyphonic.
- Mode 2: Omni On , Mono On Data=0: All channels are listenned in monophonic.
- Mode 3: Omni Off , Poly On Data=0: Basic Channel only is listened in monophonic.
- Mode 4: Omni Off , Mono On Data=M: Only MIDI channels from "Basic Channel" up to Basic Channel+M-1 are allowed in monophonic.

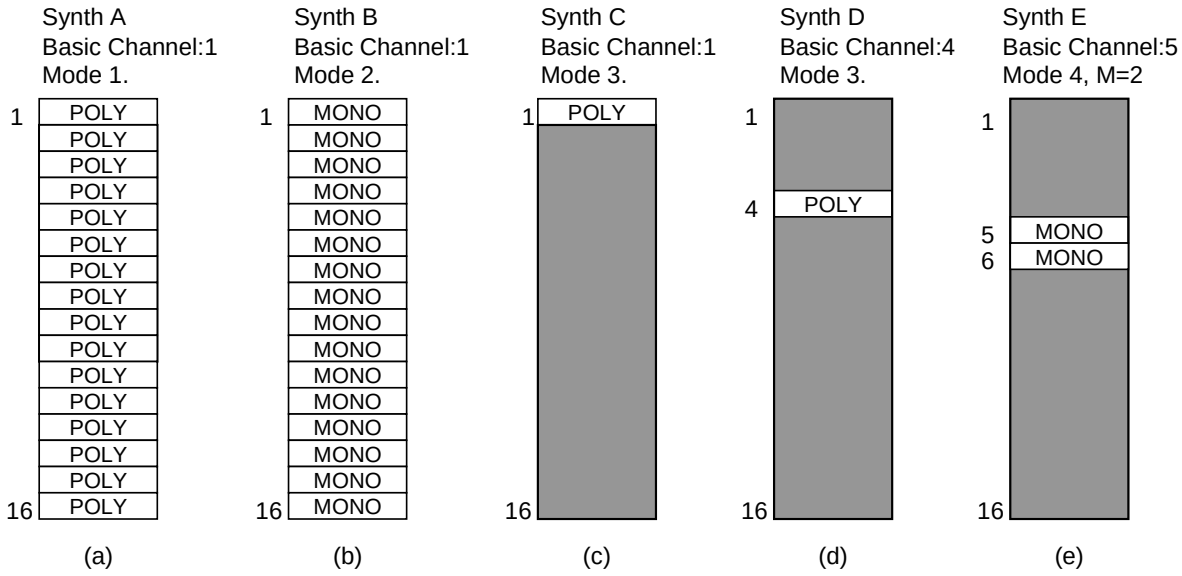


Fig. 1
Notes:
N1: mode **Omni On** (mode 1 and 2) allows to set a receiver listenning all MIDI channels. In this mode we are sure that the synthesizer is listening any MIDI channel.

- Fig 1.a shows un synthesizer A set in mode 1 on "basic channel" 1. All MIDI channels (1 to 16) are listened in polyphonic.
- Fig 1.b show a synthesizer B set in mode 2 on "basic channel 1". All MIDI channels (1 to 16) are listened in monophonic.

N2: mode **Omni Off** (mode 3 and 4) allows somes MIDI channels to be not listened. So we can use more than one receiver (C,D,E) in a manner that a MIDI channel can be listened by only one receiver at a time. To get this result we need to set each receiver on distinct basic channel.

- Fig 1.c shows a synthesizer (C) set in mode 3 on "basic channel" 1. MIDI channel 1 is the only one listened in polyphonic.

- Fig 1.d shows a synthesizer (D) set in mode 3 on "basic channel" 4. MIDI channel 4 is the only one listened in polyphonic.

N3: With **mode 4** it is easy to choose a group of continus MIDI channels (M > 1). This mode is suited to strings instruments (guitar, bandjo,..) on which a string can play only one note at a time.
Furthermore, in this mode we can use a CC to control all the M MIDI channels at the same time.

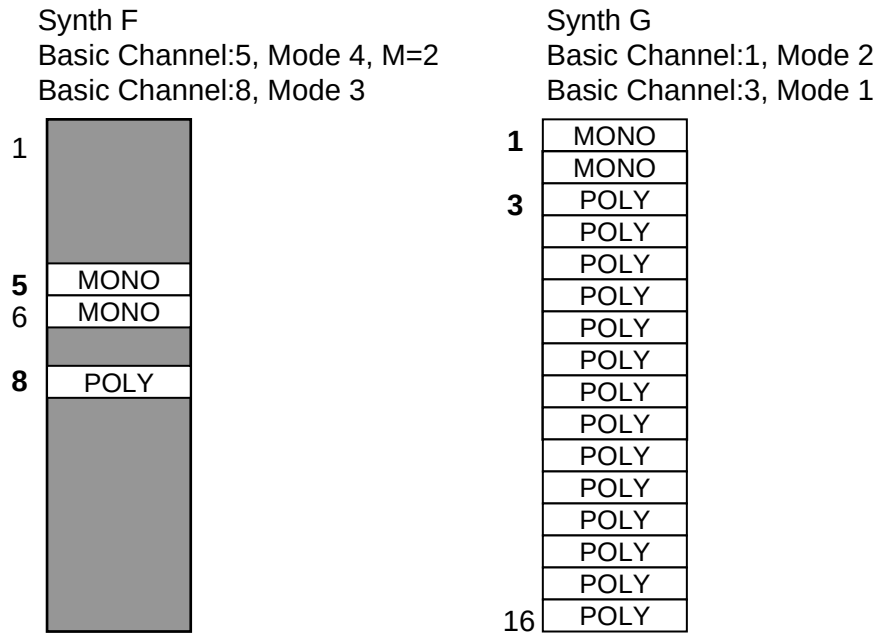
- Fig 1.e show a synthesizer (E) set in mode 4 , M=2 on basic channel 5. Only MIDI channels 5,6 are listened in monophonic.

N4: Note that with only one "Basic Channel" (Fig 1.a to Fig.1 e,) there are no possibility to have somes channels polyphonics (mode 1,3) and others channels monophonics (mode 2,4).

Note: In MIDI standard mode number are 1 based (1 to 4) but inside FluidSynth mode number are zero based (0 to 3).

2.5. Receiver with more than one "Basic Channel"

A fluidsynth instance MIDI receiver can have more than one "basic channel" (MIDI specs(1.0 v 4.2 p7)).



Each basic channel can be set in distinct mode. This is called "muti-mode".

Fig.2

fig 2.f shows a fluidsynth instance (F) with 2 "Basic Channel" in 2 distinct modes.
- Mode 4 is set on basic channel 5. MIDI channel 5 and 6 are listened in monophonic.
- Mode 3 is set on basic channel 8. MIDI channel 8 is listened in polyphonic.
- Others MIDI channels aren't listened.

2.5.1. Using Omni On (mode 1 and 2) with more than one "Basic Channel"

When a receiver have more than one basic channel (i.e BCx, BCy), a MIDI message Omni On received on Basic Channel (BCx) set the range of MIDI channels from BCx up to BCy-1 enabled .

Fig 2.g shows a fluidsynth instance (G) with 2 Basic channel 1 and 3 both in mode Omni On:
- Mode 2 is set on basic channel 1. MIDI channels 1 and 2 are listenned in monophonic.
- Mode 1 is set on basic channel 3. MIDI channels 3 to 16 are listened in polyphonic.

2.6. Poly/Mono mode API in FluidSynth

Following API hare added .Thes API are used by the new shell commands (2.7).

2.6.1. Reset basic channels: **fluid_synth_reset_basic_channels(n, BasicChannelsInfos)**

```
FLUIDSYNTH_API
int fluid_synth_reset_basic_channels(fluid_synth_t* synth,
                                     int n,
                                     fluid_basic_channel_infos_t *basicChannelInfos)
```

The function set a new list of basic channel informations in the synthesizer.
This list replace the previous list.

On input

- ***synth*** FluidSynth instance
- ***n*** number of entry in basicChannelInfos (0 to 256)
- ***basicChannelInfos*** The list of Basic channel infos to set.

If n is 0 or basicChannelInfos is NULL, the function set one channel basic at basicchan 0 in Omni On Poly (i.e all the MIDI channel are polyphonic).

Each entry in the table is a fluid_basic_channels_infos_t
-*basicchan* is the Basic Channel number (0 to MIDI channel count).
-*mode* is MIDI mode infos for basicchan (0 to 3).
-*val* is the value (for mode 3 only) (0 to MIDI channel count).

On Output

- FLUID_OK if success
- FLUID_POLYMONO_WARNING warn about entries coherence in the table.
  -Different entries have the same basic channel, an entry supersedes a previous entry with the same basic channel.
  -Val have a number of channels that overlaps the next basic channel.
  Anyway, the function restricts val to the right value.
- FLUID_FAIL,
  - synth is NULL.
  - n, basicchan or val is outside MIDI channel count.
  - mode is invalid.

Note:This API is the only one to set all the basic channels in one synth instance.
The default shell have equivalent command "*resetbasicchannels*" to set one or more basic channels (see 2.7.2).
When more than one basic channels are set, the synthesizer is said to be in multi mode.

2.6.2. Get Basic Channels count: **fluid_synth_count_basic_channels()**

Returns the number of MIDI basic channels that the synthesizer uses internally.

FLUIDSYNTH_API int fluid_synth_count_basic_channels(fluid_synth_t* synth);
Paramètres d'entrées: ***synth*** the synthesizer object
Paramètres de sortie: Count of basic channels

This API is a short version of **fluid_synth_get_basic_channels().**

Note: not implemented

2.6.3. Get basic channels: **fluid_synth_get_basic_channels()**

The function get the list of basic channel informations in the synthesizer.

```
FLUIDSYNTH_API
int fluid_synth_get_basic_channels( fluid_synth_t* synth
                                    fluid_basic_channel_infos_t **basicChannelInfos);
```

On input

- ***synth*** FluidSynth instance
- ***basicChannelInfos***
  If non NULL the function returns a pointer to allocated table of fluid_basic_channels_infos or NULL on allocation error.The caller must free this table when finished with it.

  If NULL the function returns only the count of basic channel.
  Each entry in the table is a fluid_basic_channels_infos_t
  -*basicchan* is the Basic Channel number (0 to MIDI channel count-1)
  -*mode* is MIDI mode infos for basicchan (0 to 3)
  -val is the number of channels (0 to MIDI channel count))

On Output

- Count of basic channel informations in the returned table or FLUID_FAILED if synth is NULL or allocation error.

Note: The default shell have equivalent command "*basicchannels*" to display basics channels (see 2.7.1).

2.6.4. Set basic channel: **fluid_synth_set_basic_channel(chan,mode, val)**

```
FLUIDSYNTH_API
int fluid_synth_set_basic_channel(fluid_synth_t* synth, int basicchan,int mode, int val)
```

The function changes a MIDI Channel Mode on a Basic Channel or adds a new basic channel.

On input

- ***synth*** FluidSynth instance
- ***chan*** MIDI Basic channel number (0 to MIDI channel count - 1) to set
- ***mode***
  0:OmniOn_Poly 1:OmniOn_Mono
  2:OmniOff_Poly 3 OmniOff_Mono
- ***val***: Number of monophonic channels (for Mode 3 only) (0 to MIDI channel count).

On Output

- FLUID_OK if success
- FLUID_POLYMONO_WARNING prevent about entries coherence.
  1)val of the previous basic channel has been narrowed or
  2)val have a number of channels that overlaps the next basic channel part .
  Anyway, the function does the job and restricts val to the right value.
- FLUID_FAIL,
  - synth is NULL.
  - chan or val is outside MIDI channel count.
  - mode is invalid.

Note: The default shell have equivalent command "*setbasicchannels*" to set basics channels mode (see 2.7.3).

2.6.5. Get channel mode: **fluid_synth_get_channel_mode(chan, modeInfo)**

```
FLUIDSYNTH_API
int fluid_synth_get_channel_mode(fluid_synth_t* synth, int chan,
                                 fluid_basic_channel_infos_t  *modeInfo)
```

The API function returns poly mono mode informations about any MIDI channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** any MIDI channel number to get mode (0 to MIDI channel count - 1).
- ***modeInfo***: pointer to returned mode infos Mode.
  A fluid_basic_channels_infos_t
  -basicchan, chan
  -mode is MIDI mode infos of chan:
  - bit 0: **MONO**: 0, Polyphonique; 1, Monophonique.
  - bit 1: **OMNI**: 0, Omni On; 1, Omni Off.
  - bit 2: **BASIC_CHANNEL**: 1, this channel is a Basic Channel.
  - bit 3: **ENABLED** 1, chan is listened;
    0, voices messages (MIDI note on/of, cc) are ignored on chan.

  -val number of channels in the group from basic channel (if bit 2 is set), or 0 if bit 2 is 0.

On Output
- FLUID_OK if success
- FLUID_FAIL
  -synth is NULL.
  -chan is outside MIDI channel count or modeInfos is NULL

Note: The default shell have equivalent command "*channelsmode*" to display basics channels mode (see 2.7.4).

2.7. Poly/mono modes commands in FluidSynth

2.7.1. Command to get MIDI basic channels number: **basicchannels**

basicchannels
Print the list of all MIDI basic channels informations
example: Basic channel 3 mode:2 nbr:6, Basic channel 6 mode 3: nbr:3, ..

This command uses function API fluid_synth_get_basic_channels() (2.6.3).

2.7.2. Command to replace MIDI basic channels: **resetbasicchannels**

resetbasicchannels [chan1 Mode1 nbr1 chan2 Mode2 nbr2 ….]
Set the list of MIDI basic channels with mode
This list replace any previous basic channels list.

With no parameters the function set one channel basic:
basicchan 0, mode 0 (Omni On Poly) (i.e all the MIDI channel are polyphonic).

This command uses function API_fluid_synth_reset_basic_channels() (2.6.1).

2.7.3. command to change/add MIDI basic channels : **setbasicchannels**

setbasicchannels chan1 Mode1 nbr1 [chan2 Mode2 nbr2 ….]
Change or add basic channel 1 [and 2]

-if chan is already a basic channel, its mode is changed.
-If chan is not a basic channel, a new basic channel part is inserted
between the previous basic channel and the next basic channel.
val value of the previous basic channel will be narrowed if necessary.

This command uses function API_fluid_synth_set_basic_channel() (2.6.4).

Theses commands call these functions l'API
fluid_synth_get_basic_channels() (2.6.3), fluid_synth_set_basic_channels() (2.6.4),

2.7.4. Command to read MIDI channels mode: **channelsmode**

channelsmode

Print channel mode off all MIDI channels (Poly/mono, Enabled, Basic Channel)
example

```
channel:  0, disabled
channel:  1, disabled
channel:  2, disabled
channel:  3, disabled
channel:  4, disabled
channel:  5, enabled, basic channel, mono omni off(3), nbr:  2
channel:  6, enabled, --          , mono          , --
channel:  7, disabled
```

channelmode chan1 chan2

Print only channel mode off MIDI channel chan1, chan2

These commands uses function API_fluid_synth_get_channel_mode() (2.6.5).

2.8. Part 1: implementations steps in Fluidsynth.

2.8.1. Implementation steps: Poly/mono mode API (2.6)

- API fluid_synth_reset_basic_channels()
- API fluid_synth_get_basic_channels()
- fluid_synth_set_basic_channel()
- fluid_synth_get_channel_mode()

to do	comments
	fluid_chan.h
done	Add ***mode,mode_val*** in struct _fluid_channel_t
done	*<u>/* acces to channel mode */</u>* *<u>/* SetChanMode set the mode for a MIDI basic channel */</u>* #define SetChanMode(chan,mode) \ (chan->mode = (chan->mode & ~MASKMODE) | (mode & MASKMODE)) *<u>/* GetChanMode get the mode for a MIDI basic channel */</u>* #define GetChanMode(chan) GetModeMode(chan->mode) *<u>/* IsChanMono(chan) return true when channnel is Mono */</u>* #define IsChanMono(chan) (IsModeMono(chan->mode)) *<u>/* IsChanPoly(chan) return true when channnel is Poly */</u>* #define IsChanPoly(chan) (!IsChanMono(chan)) *<u>/* IsChanOmniOff(chan) return true when channnel is Omni off */</u>* #define IsChanOmniOff(chan) (chan->mode & OMNI) *<u>/* IsChanOmniOn(chan) return true when channnel is Omni on */</u>* #define IsChanOmniOn(chan) (!IsChanOmniOff) *<u>/* IsChanBasicChannel(chan) return true when channnel is Basic channel */</u>* #define IsChanBasicChannel(chan) IsModeBasicChan(chan->mode) *<u>/* IsChanEnabled(chan) return true when channnel is listened */</u>*

	#define IsChanEnabled chan) IsModeChanEn(chan->mode) */* End of macros interface to poly/mono mode variables */*

to do	**comments**
	fluid_synth.c , fluid_chan.c
done	In fluid_chan.c - fluid_channel_init(fluid_channel_t* chan) ***mode*** and ***mode_val*** initialization.
done	In fluid_synth.c - fluid_synth_settings() settings registering.
done	In fluid_synth.c – new_fluid_synth() Reading default settings (2.2)

to do	**comments**
	synth.h
done	/* API: Poly mono mode */ */* Macros interface to poly/mono mode variables */* enum PolyMonoMode { OMNION_POLY, /* MIDI mode 0 */ OMNION_MONO, /* MIDI mode 1 */ OMNIOFF_POLY, /* MIDI mode 2 */ OMNIOFF_MONO, /* MIDI mode 3 */ MODE_NBR }; */* bit mode */* #define MONO 0x01 */* b0, 0: poly on , 1: mono on */* #define OMNI 0x02 */* b1, 0: omni on, 1:omni off */* #define MASKMODE (OMNI|MONO) #define BASIC_CHANNEL 0x04 */* b2, 1: channel is basic chanel */* #define ENABLED 0x08 */* b3, 1:channel is listened */* */* access to mode */* #define GetModeMode(mode) (mode & MASKMODE) #define IsModeMono(mode) (mode & MONO) #define IsModeBasicChan(mode) (mode & BASIC_CHANNEL) #define SetModeBasicChan(mode) (mode |= BASIC_CHANNEL) #define ResetModeBasicChan(mode) (mode &= ~ BASIC_CHANNEL) #define IsModeChanEn(mode) (mode & ENABLED) #define SetModeChanEn(mode) (mode |= ENABLED) #define ResetModeChanEn(mode) (mode &= ~ENABLED)
done	struct _fluid_basic_channel_infos_t; typedef struct _fluid_basic_channel_infos_t fluid_basic_channel_infos_t;)
done done done done	function declaration fluid_synth_get_basic_channels() function declaration fluid_synth_reset_basic_channels() function declaration fluid_synth_get_channel_mode() function declaration fluid_synth_set_basic_channel()

to do	comments
	fluid_synth.c, **fluid_synth_polymono.c**
done	function **fluid_synth_get_basic_channels()**
done	function **fluid_synth_set_basic_channel_LOCAL()** Using fluid_synth.c - fluid_synth_all_notes_off_LOCAL() that needs to declared non static to get scope in module fluid_synth_mono.c.
done	function **fluid_synth_reset_basic_channels()**
done	function **fluid_synth_get_channel_mode()**
done	function **fluid_synth_set_basic_channel()**

2.8.2. Implementation steps: Poly/mono mode commands (2.7)

to do	comments
	fluid_cmd.c
done	add entry in fluid_commands[ ]
	fluid_cmd.h
done	add functions declaration

to do	comments
	fluid_synth_polymono.c
done	command **basicchannels ,** fluid_handle_basicchannels()
done	command **resetbasicchannels ,** fluid_handle_resetbasicchannels()
done	command **channelsmode** , fluid_handle_channelsmode()
done	command **setbasicchannels,** fluid_handle_setbasicchannels()

2.8.3. Inplementation steps: MIDI mode messages (2.3)

to do	comments
	fluid_synth.c
done	In fluid_synth.c – fluid_synth_cc_LOCAL() uses fluid_synth_mono.c - fluid_synth_set_basic_channel_LOCAL() that needs to be delcared external in fluid_synth.c.
done	test non basic MIDI channel
	initial state: mode:3, nbr:2
done	cc POLY_ON, val:0 --->mode:2,nbr:1
done	cc POLY_OFF, val:0--->mode:3,nbr:16
done	cc POLY_OFF, val:3--->mode:3,nbr:3
done	cc OMNI_ON, val:0 ---->mode:1,nbr:16
done	cc OMNI_OFF, val:0 --->mode:3,nbr:1
	Etat initial: mode:2, nbr:1
done	cc POLY_ON,val:0 ---->mode:2,nbr:1

done	cc POLY_OFF, val:0--->mode:3,nbr:16
done	cc POLY_OFF, val:5--->mode:3,nbr:5
done	cc OMNI_ON, val:0 ---->mode:0,nbr:16
done	cc OMNI_OFF,val:0 --->mode:2,nbr:1
	Etat initial: mode:0, nbr:16
done	cc POLY_ON, val:0 --->mode:0,nbr:16
done	cc POLY_OFF, val:0--->mode:1,nbr:16
done	cc OMNI_ON, val:0 ---->mode:0,nbr:16
done	cc OMNI_OFF,val:0 --->mode:2,nbr:1
	Etat initial: mode:1, nbr:16
done	cc POLY_ON, val:0 --->mode:0,nbr:16
done	cc POLY_OFF, val:0--->mode:1,nbr:16
done	cc OMNI_ON, val:0 ---->mode:1,nbr:16
done	cc OMNI_OFF,val:0 --->mode:3,nbr:1

2.8.4. ignoring MIDI messages on dsabled MIDI channels

to do	comments
	fluid_synth.c
	Following API aren't dependant of channel state 'enabled'
	FLUIDSYNTH_API int fluid_synth_sysex(fluid_synth_t *synth, const char *data, int len, char *response, int *response_len, int *handled, int dryrun);
	FLUIDSYNTH_API int fluid_synth_get_channel_info (fluid_synth_t *synth, int chan, fluid_synth_channel_info_t *info);
	enum fluid_midi_channel_type { CHANNEL_TYPE_MELODIC = 0, CHANNEL_TYPE_DRUM = 1 }; int fluid_synth_set_channel_type(fluid_synth_t* synth, int chan, int type);
	Following API are enabled only when channel is enabled
done	FLUIDSYNTH_API int fluid_synth_noteon(fluid_synth_t* synth, int chan, int key, int vel);
done	FLUIDSYNTH_API int fluid_synth_noteoff(fluid_synth_t* synth, int chan, int key);
done	FLUIDSYNTH_API int fluid_synth_cc(fluid_synth_t* synth, int chan, int ctrl, int val);
done	FLUIDSYNTH_API int fluid_synth_get_cc(fluid_synth_t* synth, int chan, int ctrl, int* pval);
done	FLUIDSYNTH_API int fluid_synth_all_notes_off(fluid_synth_t* synth, int chan);
done	FLUIDSYNTH_API int fluid_synth_all_sounds_off(fluid_synth_t* synth, int chan);
done	FLUIDSYNTH_API int fluid_synth_channel_pressure(fluid_synth_t* synth, int chan, int val);
done done	FLUIDSYNTH_API int fluid_synth_pitch_bend(fluid_synth_t* synth, int chan, int val); FLUIDSYNTH_API int fluid_synth_get_pitch_bend(fluid_synth_t* synth, int chan, int* ppitch_bend);
done done	FLUIDSYNTH_API int fluid_synth_pitch_wheel_sens(fluid_synth_t* synth, int chan, int val); FLUIDSYNTH_API int fluid_synth_get_pitch_wheel_sens(fluid_synth_t* synth, int chan, int* pval);
done	FLUIDSYNTH_API int fluid_synth_program_change(fluid_synth_t* synth, int chan, int program);
done	FLUIDSYNTH_API int

	fluid_synth_bank_select(fluid_synth_t* synth, int chan, unsigned int bank);
done	FLUIDSYNTH_API int fluid_synth_sfont_select(fluid_synth_t* synth, int chan, unsigned int sfont_id);
done	FLUIDSYNTH_API int fluid_synth_unset_program (fluid_synth_t *synth, int chan);
done	FLUIDSYNTH_API int fluid_synth_get_program(fluid_synth_t* synth, int chan, unsigned int* sfont_id, unsigned int* bank_num, unsigned int* preset_num);
done	FLUIDSYNTH_API int fluid_synth_program_select(fluid_synth_t* synth, int chan, unsigned int sfont_id, unsigned int bank_num, unsigned int preset_num);
done	FLUIDSYNTH_API int fluid_synth_program_select_by_sfont_name (fluid_synth_t* synth, int chan, const char *sfont_name, unsigned int bank_num, unsigned int preset_num);
done	FLUIDSYNTH_API int fluid_synth_program_reset(fluid_synth_t* synth); used by fluid_synth_program_change()
done	(fluid_channel_reset()) Add a channel basic set in mode Omni On Poly ignoring settings

2.8.5. MIDI CC MIDI global

to do	comments
	\fluidsynth-1.1.6\src\synth\fluid_chan.h
done	adding macro #define GetChanModeVal(chan) (chan->mode_val)
	\fluidsynth-1.1.6\src\synth\fluid_synth.c
done done	adding in fluid_synth_cc(fluid_synth_t* synth, int chan, int num, int val) test: Ok

3. Part 2:Polyphonic/monophonic behavior inside Fluidsynth

This part describes the behavior of a monophonic instrument (like MIDI Wind Controller) when played by a musician (3.2).
This part describes the behavior of a polyphonic instrument (like keyboard) when played by a musician (3.3).

The synthesizer (receiver) at the other side needs to behave correctly without knowledge of the intrument type (transmitter) connected on its input. This is how FluidSynth behaves.

3.1. Part 2: Tutorial examples- understanding monophonic behavior

This chapter describes useful examples to learn and understand what is poly/mono mode
Example files are directly executable on the console using the source command.

>source command-example-file

3.1.1. How to set a MIDI channel in mono mode ?

To be able to play monophonically a MIDI channel must be set in mono mode. This can be done by setting basics channels (see 2.1 for détails.)
This can be done also by using legato pedal On (see 3.1.2).

3.1.2. Why using legato pedal On/Off ?

When a channel is in poly mode, it can be set temporarily in mono mode during performance playing by deperessing legato pedal (cc 68d). When legato is On, the channel reacts as if it was set in mono mode. That means that if the musician is playing legato the channel will react legato (because the channel is in mono state). When the musician release legato pedal, the channel restore in the state prior legato On. In others words legato On/Off is useful only when playing on MIDI polyphonic controller.

When playing on MIDI monophonic controller (any electronic wind instrument, or electonic valve instrument) normally this kind of instruments send CC legato On/Off.

In summary when a channel receives CC legato On/Off it will be able to reacts monophonically regardless the poly/mono mode in witch it is set.
Legato On/Off is a real time performance CC to allows a polyphonic instrument to be played like monophonic instrument.
This doesn't mean that legato On/Off supersede the functionnality offers by 'basic channels' (see 2.2).

When a channel is in mono mode (set by basic channels 2.2), this channel is always able to play legato (if the musician plays legato) regardless of legato pedal.

See 3.4.10 for détails.

3.1.3. How reacts FluidSynth when the musician plays legato or staccato ?

When a channel is mono or legato is On, the channel reacts to the musician playing.
That means that if the musician is playing staccato the channel will play staccato.
If the musician plays legato , the channel will react legato.

3.1.4. What are legato mode ?

Legato mode are only used when the musician plays legato on a channel that is in mono state (i.e the channel is mono or CC legato is On) (see 3.1.2).

Legato mode define the way the note transition n1,n2, is articulated on n2On. There are 5 legato modes. When a channel is set in a mode, articulations betwween note transition always make use of this mode.

When the musician play staccato, the notes are articulated normally (as defined by the soundfont preset) regardless of legato mode.

Examples
Presets numbers in Legato_demo.sf2:
66 Tenor Sax 56:Trumpet 71:Clarinet
70: Bassoon 73:Flute 74:Alto Recorder Yamaha
The following examples use preset Tenor Sax (66)

legato mode 0: **source leg_0.txt**
legato mode 1: **source leg_1.txt**
legato mode 2: **source leg_2.txt (see remark)**
legato mode 3: **source leg_3.txt (see remark)**
legato mode 4: **source leg_4.txt (see remark)**

Remark:
- Note the difference between mode 0 vs mode 1. Mode 0 is more percussive than mode 1 due to fast release (mode 0) vs normal release (mode 1). This is more obvious using flute.
- Dependant of the switching preset zones, controled both by velocity range and key range , legato mode (2 to 4) can use temporarily the same articulation than legato mode 1. This is audible in the examples above. On noteOn, each time there is a switch ("velocity range" or "key range"), the note restarts the attack. In all exemples velocity is the same for all the notes, so there is no "velocity range" switching".

- If you want to change the preset you can edit the file leg_x.txt and modify the **prog** command.

See 3.7 for détails.

3.1.5. What are breath mode ?

Breath mode allows a keybordist to control dynamic with the help of a Breath controller.

This functionnality is useful only in case of no CC Breath modulator inside the Soundfont.
It is a quick way to try the effect of Breath Controller (please see the important note below).

This patch gives FluidSynth the possibility to set a Default *Breath To InitialAttenuation* inside FluidSynth with the help of a shell command.
The command allows to set a cc Breath modulator to replace the default "*velocity to initial Attenuation*" modulator for a channel, independently when played polyphonic or monophonic.

Important:
- Any identical modulators in the Soundfont will supersede the default breath modulators set by breath mode commands.
- After you have done your try , for portability and Soundfonts sharing, it is important to add the necessary modulators in the Soundfonts using the appropriate editor.

Breath mode allows also the musician to synchonize noteOn/noteOff using the breath controller.
(see 3.4.4 for détails.)

3.1.6. What is portamento ?

Portamento is a smooth pitch sweep produced on noteOn. The sweep start from a 'from key' note to reach noteOn note smoothly.

Examples
Presets numbers in Legato_demo.sf2:

66 Tenor Sax	56:Trumpet	71:Clarinet
70: Bassoon	73:Flute	74:Alto Recorder Yamaha

The following examples use preset Trumpet(56). Note that accoustic Trumpet doesn't not allow portamento !. So this demo is exotic !

legato mode 0, with portamento: **source leg_port_0.txt**
legato mode 1, with portamento: **source leg_ port_1.txt**
legato mode 2, with portamento: **source leg_ port_2.txt**
legato mode 3, with portamento: **source leg_ port_3.txt**
legato mode 4, with portamento: **source leg_ port_4.txt**

Remark:
- Dependant of the switching preset zones, controled both by "velocity range and key range" , legato mode (2 to 4) can use temporarily the same articulation than legato mode 1. This is audible in the examples above because of the portamento presence. On noteOn, each time there is a switch ("velocity range" or "key range"), the note restarts the attack. In all exemples velocity is the same for all the notes, so there is no "velocity range" switching".
- If you want to change the preset you can edit the file leg_x.txt and modify the **prog** command.

See 3.4.9 for détails.

3.1.7. What are portamento mode ?

With portamento mode you choose the situation in wich portamento occurs. This situation (i.e this mode) is set with portamento mode commands (see 3.9 for détails.)

3.1.8. What about sustained note in monophonic mode ?

When in mono mode, note can be sustained (by sustain or sostenuto pedal) like in poly mode.
While a note is sustained, playing staccato always release previous sustained note and than sustain the new note. The result sounds like legato (mode 1).

3.2. monophonic MIDI Wind Controller instrument

This chapter describes MIDI Wind Controller behaviour supplied by Louis B.

- see starting thread [Fluid-dev] Help about MIDI Wind Controller behavior 30 Apr 2015

With the help of Louis B, it was easy to implement the monophonic behavior in this patch. Thanks to Louis B.

Many thanks to Ben Gonzales for informations about monophonic accoustic instruments and electronic winds instruments behaviors.

3.2.1. Basic behavior

Starting sending note,dynamic control and ending sending note is done when blowing in breath controller.

- No notes are sended when the musician doesn't blow.
- Playing start when blowing start. A MIDI noteOn is sended with a velocity value coming from Breath value. Notes value depends of the pressed key.
  While breath continue, the musician can change key. This is a legato manner playing (see 3.2.3).
  There is a note change on each key change, 2 consecutives MIDI messages are sended(see R1).
  {noteOn n2, noteOff n1}, {noteOn n3, noteOff n2}, (voir R2)...
  The noteOn velocity is the current breath value.
  Legato passage can be in ascending {noteOn 2, noteOff n1},{noteOn n3, noteOff n2} or descending {noteOn 2, noteOff n3} (see R3) order.
- Playing stops when the musician stops blowing; a MIDI noteOff is sended with the note which was played .
- Between starting end ending blowing , if the musicien keeps the same key, only one note is sended (noteOn n, noteOff n). Doing this several times is a staccato playing manner (3.2.2)

Remarks:
R1: noteOff presence is a MIDI standard recommendation that says that a noteOn must be send for each noteOn sended.

We remark that to reproduce a legato passage, typical synthesizer behavior is to use the voices of the previous note.

R2: We note also that during a legato passage, a MIDI noteOn can be sended while the musicien release a key.

R3: We remark also that when receiving noteOff messages a synthesizer needs only care on the last noteOff received and ignore previous one.

3.2.2. Playing staccato

To start and stop a note n1, musician start and stop blowing.

When musician blows in breath controller, the MIDI Wind Controller sends CC Breath followed by noteOn n1.

- daten1_on: CC breath 2(MSB), data > 0
- daten1_on: CC breath 34(LSB), data >0
- daten1_on: noteOn n1, vel = databreathMSB

When the musician release blowing, the MIDI Wind Controller sends noteOff n1 followed by CC Breath.

- daten1_off: noteOff n1, vel=0
- daten1_off: CC breath 2(MSB),data =0
- daten1_off: CC breath 34(LSB), data= 0

MIDI noteOn,noteOff stream is framed by CC Breath. Velocity of note is value of MSB CC breath.

3.2.3. Playing legato: n1,n2,n1

To get a legato result the musician plays a note n1 by blowing and plays an other key n2 keeping blowing.

Playing note n1

- daten1_on: CC breath 2(MSB), data > 0
- daten1_on: CC breath 34(LSB), data >0
- daten1_on: noteOn n1, vel = databreathMSB

Playing note n2 legato with n1. a noteOn n2 is sended followed by a noteOff n1 (see R3). noteOn n2 is preceded by CC legato On (see R1,R2).

- daten2_on: CC legato On
- daten2_on: noteOn n2, vel = current data breathMSB
- daten2_on: noteOff n1, vel=0

R1: The MIDI Wind Controller detects a legato playing because the musicien continue to blow at n2 key time while n1 was still pressed. This wan't not the case at n1 time.

R2: The MIDI Wind Controller detects the start of a legato passage and sends CC legato On which informs the synthesizer that it needs to interpret n2 legato with the more recent note received (i.e n1). So even if the synthesizer is in polyphonic mode (see 3.5), it can reacts has if it was in monophonic mode (see 3.4.2)..

R3: a noteOff n1 is sended for each noteOn sended (standard MIDI).
The synthesizer can ignore this message.

The musician continue legato playing, keeping blowing and playing key n3 (legato n2,n3) .

- daten3_on: noteOn n3, vel = current data breathMSB
- daten3_on: noteOff n2, vel=0 (voir R3)

The MIDI Wind Controller detects a running legato n2,n3. So it idoesn't send unnecessary cc legato On. Receiving n3, the synthesizer remember of a running legato state, so it reacts legato n2,n3 (same as R2).
Remark R3 is relevant.

The musician can continue legato playing, keeping blowing and releasing key n3 (legato n3,n2)
Remarks R1,R3 sare still relevant.

- daten3_off: noteOn n2, vel = current data breathMSB
- daten3_off: noteOff n3, vel=0 (see R3)

The MIDI Wind Controller detects a legato playing n3,n2 because at n3 release time, key n2 is still pressed.
The synthetizer reacts the same way.

When the musician release breath a noteOff is sended (R4)

- daten2_off: noteOff n2, vel=0
- daten2_off: CC breath 2(MSB),data =0
- daten2_off: CC breath 34(LSB), data= 0

If the musician restarts blowing , has key n1 and n2 are still pressed ,the MIDI Wind Controller send a noteOn n2 that could be played legato with n1 at musician desire.

- daten2_on: CC breath 2(MSB), data > 0

- daten2_on: CC breath 34(LSB), data >0
- daten2_on: noteOn n2, vel = databreathMSB

The musician can continue legato playing, by breath maintain and releasing key n2
- daten2_off: noteOn n1, vel = current data breathMSB
- daten2_off: noteOff n2, vel=0 (voir R3)

When the musicien release breath a noteOff is sended (R4) eventualy followed by legato Off (R5) if legato was running.
- daten1_off: noteOff n1, vel=0
- daten1_off: CC legato off
- daten1_off: CC breath 2(MSB),data =0
- daten1_off: CC breath 34(LSB), data= 0

R4: Synthesizer must play this event, it doesn't ignore it has it does in R3.
The MIDI Wind Controller detects a legato ending because it knows that the played note was the last pressed note.

3.3. Polyphonic controller (Keyboard)

3.3.1. basic behavior

Each key is independant. Notes can be played simultaneously.
A MIDI noteOn is send when the musician press a key and a MIDI noteOff is send when the key is released.

3.3.2. Playing staccato

Musician can play staccato. MIDI stream messages (noteOn/noteOff) are the same that with MIDI Wind Controller

3.3.3. Playing legato

A polyphonic MIDI controller doen't not detect legato playing. So it it very difficult for a musician (even for a skilled one) to get a legato result when playing legato.
So, a Polyphonic controller doesn't send MIDI legato On/Off automatically (has does MIDI Wind Controller see 3.2).

3.4. Synthesizer monophonic behavior

This chapter describes the synthesizer behavior when it receive MIDI messages on monophonic channel. The synthesizer must behave the same regardless which MIDI controller (monophonic,polyphonic) is connected on its input.
- polyphonic controller (keboard) to monophonic channel (see 3.4.2).
- CC Breath controller to) to monophonic channel (see 3.4.4).
- Polyphonic controller (MIDI Wind Controller) to monophonic channel (see 3.4.6) .

3.4.1. Input controller (mono/poly)

The synthesizer must behave the same regardless which MIDI controller (monophonic,polyphonic) connected on its input. This must be true regadless the playing manner (staccato or legato). Chapter 3.4.2 gives the algorithm when polyphonic controller is on input.

3.4.2. polyphonic controller (keyboard) to monophonic channel.

When playing staccato on keyboard, synthesizer behavior is the same regardless channel mode Poly/Mono.
When playing legato, it is different. The synthesizer needs to remember of the notes who belongs to a legato passage from the 1st note. This memory is necessary to allow descendant playing legato.

3.4.3. Legato playing detector – the monophonic list

This list remember the notes in playing order. It allows an easy automatic detection legato passage when it is played on a keyboard.

- (a) On noteOn ***n2,*** if a note ***n1*** is running, there is a legato detection with the running note *n1* (with or without portamento from *n1* to *n2*).
- (b) On noteOff of the running note ***n2,*** while a previous note is running, there is a legato detection from n2 to n1 (with or without portamento from *n2* to *n1*).

Each MIDI channel have a monophonic list.

3.4.4. CC Breath controller to monophonic channel .

The MIDI breath controller on MIDI Wind Controller allows fluid dynamic control beetween notes of a legato passage.
Furthermore starting and stopping notes is triggered via the breath controller.
This possibility allows differents articulations inside a legato passage.

Consequently , it is very important that the synthesizer reacts to CC Breath. (3.4.5).

polyphonic controller (keyboard) on input
When the musicien play on a keyboard, the dynamic is controled only at noteOn time but not between noteOn. So a keyboard suffers of a lack of dynamic control.
The chapter 3.4.5 is a proposal to bring CC breath.

3.4.5. How FluidSynth can play CC Breath controller

It is up to the soudfont preset designer to add CC Breath support. If a CC "Breath to attenuation" modulator is present in the soundfont, Fluidsynth synthesizer will play it. It is the right way as it allows preset sharing.

To get FluidSynth able to play CC Breath, it is necessary that the preset have the following properties:

- 1)Cancel the effect of the default modulator *Velocity To Initial Attenuation*
  This can be done by adding a modulator *Velocity To Initial Attenuation* with amount value to 0.
  This is not mandatory, it just useful for keyboardist who want to control dynamic with only a breath controller (but not with both key velocity + breath controller).
  For MIDI Wind Controller player, as far the MIDI Wind Controller put breath data in velocity data (which is normally the case), step (1) is unnecessary.

- 2) Adding a *CC Breath To Initial Attenuation* modulator with a amount of 960 cB.
  This amount value is dependant of the synthesizer "dynamic range" capability. Actually FluidSynth range is 960 cB as the internal audio engine generates 16 bit samples.

The best is to set this 2 modulators in the Local Zone (Intrument Zone).
A preset define a sound, so it is logical that those modulators are set in the SoundFont.

However, this patch gives FluidSynth the possibility to set a Default *Breath To InitialAttenuation* inside FluidSynth with the help of a shell command.
This is useful in case of no CC Breath modulator inside the Soundfont.
The command allows to set a cc Breath modulator to replace the default "*velocity to initial Attenuation*" modulator for a channel, independently when played polyphonic or monophonic.

setbreathmode chan 1 | 0 1| 0

Examples

- No default CC Breatth To InitlAttenuation for MIDI Channel 2
  setbreathmode 2 0 0

- MIDI channel 2: Breath modulator for poly mode only.
  setbreathmode 2 1 0

- MIDI channel 2: Breath modulator for mono mode only.
  setbreathmode 2 0 1

- MIDI channel 2: Breath modulator for both mono and poly mode.
  setbreathmode 2 1 1

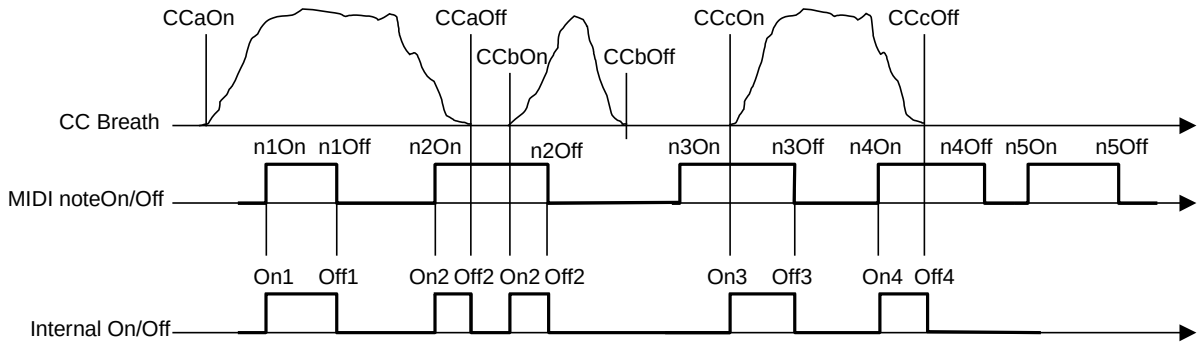
This patch brings a new options called "Breath noteOn/noteOff". This option is the 4th parameter of setbreathmode command

It allows the breath controller (MSB) to trigger noteoff/noteon on the running note. If you are a wind player and you want to play on a keyboard, you would appreciate similar behaviors than electronic wind instruments. So this option is only efficient if the MIDI input device is a keyboard. If the MIDI input device is an electronic wind instrument, this option does nothing as this intrument naturally sends noteOn and noteOff when the player starts and stop blowing.
Example on MIDI channel 4

setbreathmode 4 0 1 1

Parameter 3 is 1 to enable " Breath modulator " for mono mode
Parameter 4 is 1 to enable "breath noteOn/Off" for mono mode only

Summary
- "Breath sync noteOn/noteOff" option works only for mono mode (or poly mode with legato On).
- parameter 3 is not mandatory but is is a natural choice for a wind player.
- Breath sync is useful only for MIDI keyboard on input.
- Playing on the keyboard will triggers noteOn/noteOff only if the player blow.
- Starting /stopping blowing will triggers noteOn/noteOff only if the player hold a key.
- During blowing, the musician can plays legato or not with the keyboard.
- In others words this option allows to play staccato with a Breath controller.



The figure illustrate the "Breath sync noteOn/noteOff" option behavior.

3.4.6. Monophonic controller (MIDI Wind Controller) to monophonic channel.

We remark that the behavior of the algorithm that allows a channel to react monophonically (when played by a polyphonic controller (keyboard) (see 3.4.2) works also when the input controller is monophonic (MIDI Wind Controller). FluidSynth reacts the same way regardless type of input controller.

3.4.7. Using Sustain / Sostenuto in monophonic mode

A monophonic channel note can be hold by Sustain or Sostenuto, the same way as for polyphonic channel.

Note: On noteOn n1 if a previous monophonic note *np* is held by Sustain (or Sostenuto) this note *np* is released.

3.4.8. Useful MIDI CC in monophonic mode

Controller	number		Value	Monophonic only.
Portamento time MSB	5d	05h	0 -127	Mono/Poly (see Portamento On/Off) and Portamento Control.
Portamento time LSB	37d	25h	0 -127	Mono/Poly (see Portamento On/Off) and Portamento Control
Portamento off/on	65d	41h	<= 63, >=64	Mono only
Legato off/on	68d	44h	<= 63, >=64	Mono/Poly
Portamento control	84d	54h	0 -127	Poly/Mono
Hold 2	69d	45h		

Hold 2 allows to freeze current ADSR generator value (page 69 MIDI specifications).

This patch supports CC in bold only.

3.4.9. Portamento On/On

When pedal is On, pitch sweep is enabled in both poly or mono mode.
the speed of sweep is instructed by Portamento time (MSB,LSB) in ms. If portamento time is 0 the sweep is not perceptible.
See 3.9 for details.

3.4.10. Legato On/On

If a channel is set in poly mode, it can't play monophonically. In this situation the musician can turn this channel in monophonic state by depressing legato pedal. During the time the pedal is hold, this channel behaves monophonically as if it was set in mono mode.
That means that the channel is able to play legato if the musician plays legato.
A musician plays legato n1,n2 when he plays n2 without releasing n1.

Remarks:

- In Fluidsynth, on poly channel , if note n1 is hold before legato is depressed, when n2 is played On, legato passage begin with note n1. This way, if the MIDI input device is a keyboard, the synthesizer behaves the same that if the input device was an electronic wind controller intruments. Wind players would appreciate identic behaviors.
- A channel set in mono mode (by basic channels commands(2.2), or CC poly/mono (2.3)) , ignore legato pedaling. This channel is able to play legato if the musician plays legato.

3.4.11. CC Portamento Control (PTC)

CC portamento control is a MIDI specification.
When CC portamento is received, the next note is played portamento regardless of Portamento pedal.
The portamento fromkey note is given by the value of this CC. This work in Mono or Poly mode.

In both mode, when the value of the CC coincide with a possible running note (legato in mono and poly).
The note (tokey) is played using the voices of the running note.
In Poly this produce a localized legato mono transition in the context of polyphonic playing.
For exemple while the musician hold C major chord (C,E,G) if CC PTC with value G is received, when the musician play A note, the G voices of the chord are stolen to play a portamento from G to A.

When portamento sweep is finished the chord is C E A.

If the value of PTC doesn't coincide with a running note, with the same chord example (C,E,G), no chord voices are stolen. When portamento sweep is finished the chord is C E G A.

3.4.12. CC Global to control all channels at the same time (Mode 3) (OmniOff- Mono).

CC global is a MIDI specification.
If a there is a basic channel in mode 3, it is possible to send a CC for all MIDI Channel that belong to this group. The CC is called a grobal CC. To be accepted as global, the CC must be send on a global channel one below the basic channel and that grobal channel must not belong to another group otherwise the CC is considered as a normal CC (send only to one channel).

In Fluidsynth It is always possible to have more than one basic channel with the use of CC grobal in perspective (see 2.5).

3.5. Polyphonic Channel behavior

When playing on a polyphonic channel, the musician can use the legato pedal to enter the channel in monophonic mode.

3.6. monophonic list implementation

The monophonic list is a circular list. The notes are added by order of arrival. There are 10 entry

The methods are
- Initialization (3.6.1).
- add a note: **fluid_channel_add_monolist()**
- search a note: **fluid_channel_search_monolist()**
- remove a note: **fluid_channel_rem_monolist()**
- keep only last note played: **fluid_channel_keep_lastnote_monolist()**
- Set only one note in the list: **fluid_channel_set_onenote_monolist()**

3.6.1. List initialisation

- iFirst= 1 /* First note index to begin a search */
- iLast = 0 /* index used during recent adding */
- nNotes = 0; /* actual number of notes */
- PrevNote = InvalidNote /* Previous note : invalid at initialisation time*/

3.7. Legato modes

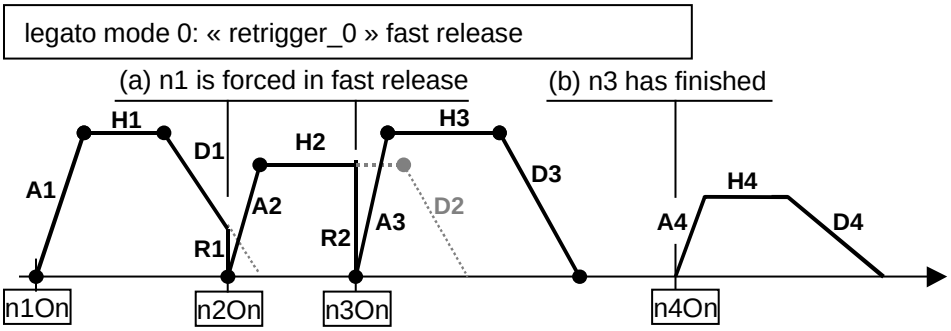
This chapter describes legato modes. Legato modes instructs a channel on how to articulate the notes of a legato passage. This triggering mode are interresting with a keyboard on input witch have velocity or a preset with true adsr volume articulations (i.e with attack decay and sustain not confused). These modes have little or no effect on "flat" adsr enveloppe.

These modes articulate the notes follwing the 1st note differently.
For example: let n1,n2,n3,n4,.....
n1 is the 1st note of a legato passage. It is played normally.
n2,n3,n4 are articulated differently than n1. ***Legato mode*** instructs the type of articulations.

There are 5 modes numbered 0 to 4. The more numbered have the more legato articulation contribution. Mode 0 have the less legato articulation contribution (it sound more percussive on attack).

The legato mode can be set by API (3.7.7) or commands (3.7.9).

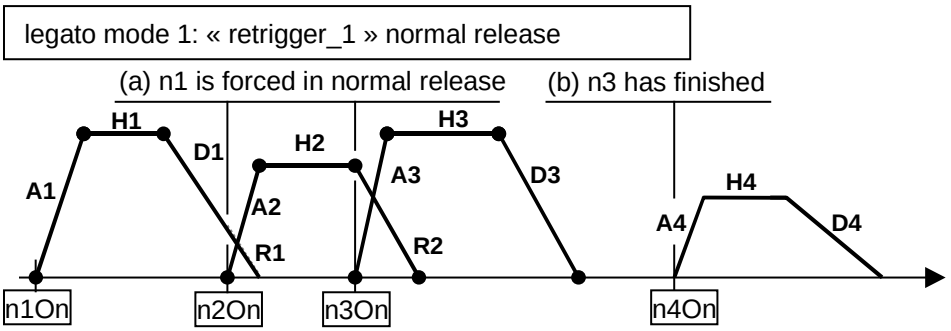
3.7.1. Mode 0: "retrigger_0" (fast release)



- (Mode 0) On noteOn n2,n3....The previous note is released quickly and the new note is started normally.
  The musician gives velocity/breath on each note (n1,n2,n3).
  This mode is called "retrigger" because adsr are restarted from 0. This is why the legato perception is diminished.

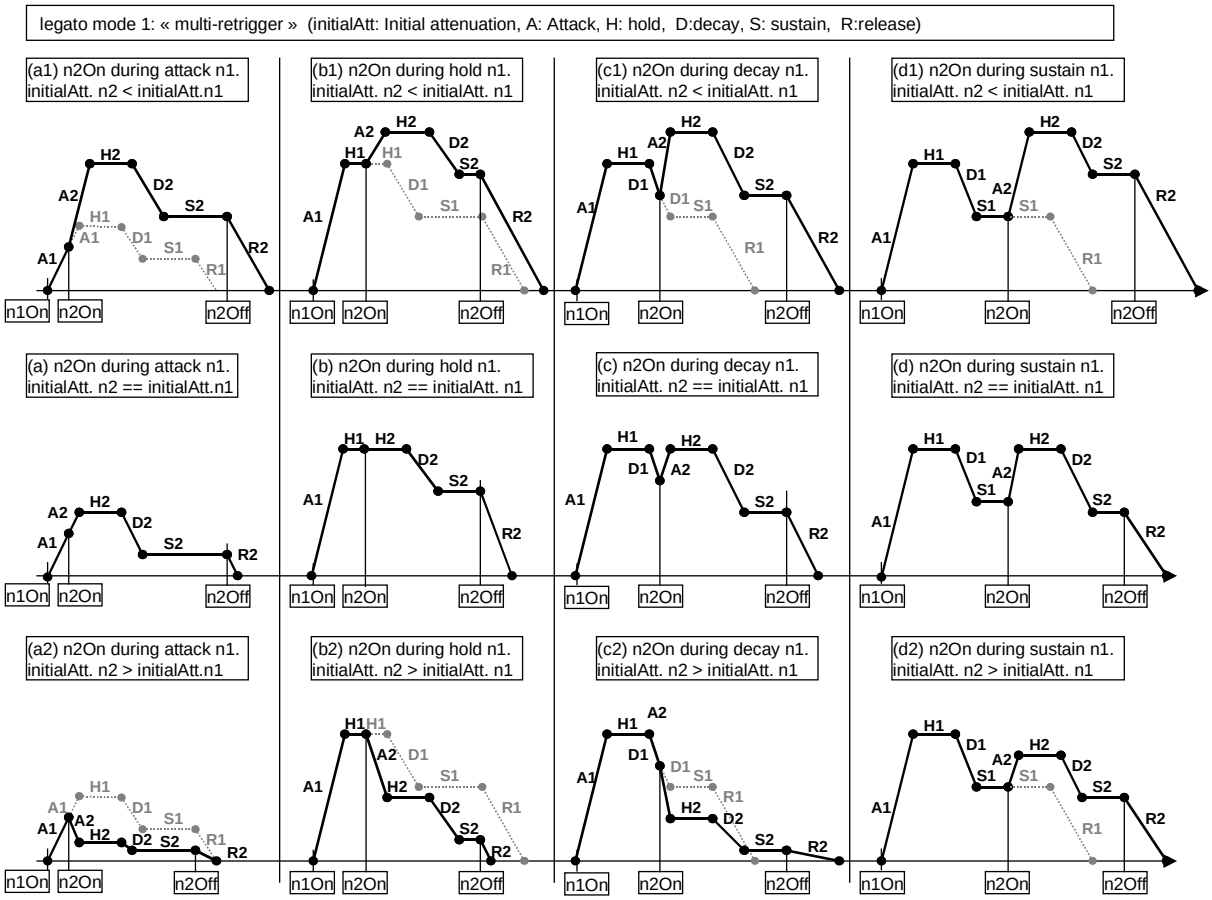
  Remark: for example,this mode is adequat for playing a lot of trills.
  If the musician don't want playing trill he just needs to release previous note before releasing current note .

3.7.2. Mode 1: "retrigger_1" (normal release)



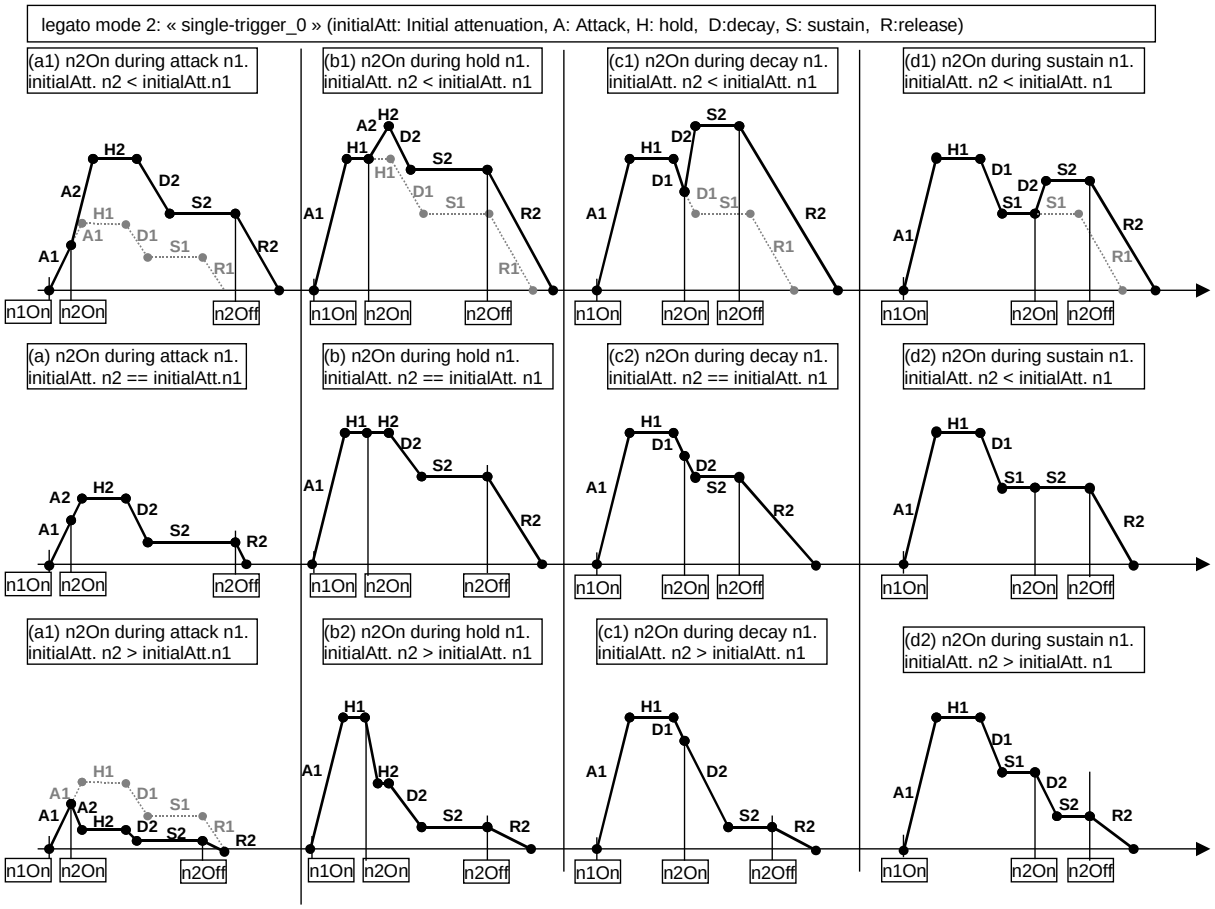
This mode est similar to previous mode 0, but previous note are not released quickly. They release normally as release time is not changed. If relase is not too fast there is a cross fading between release of previous note and attack of the current note. This "cross fadind" offers a more legato perception than mode 0.

3.7.3. Mode 2: "multi-retrigger"



(Mode 1) On noteOn n2,n3,... adsr are forced in attack section but keeping current envelope value.
The attack section is reshaped by current velocity.
The musician gives velocity/breath on each note (n1,n2,n3).
This mode is called multi-retrigger because adsr generators are retriggered in attack section.
Depending on the envelope shape, this mode is more legato than mode 0.

3.7.4. Mode 3: "single-trigger_0"



- (Mode 2) On noteOn n2,n3,... adsr stay in the current section keeping the current envelope value.
  The current section is re-shaped by the current dynamic.
  The musician gives velocity/breath on each notes (n1,n2,n3).
  This mode is called single-trigger because the envelopes are triggered one time (on n1On).
  If the enveloppe has a sustain (above 0), this mode have more legato effect than mode 1.

3.7.5. Mode 4: "single-trigger_1"

- (Mode 3) This is the normal mode. This mode is similar to mode 2 ,but the adsr envelope are not shaped. As this mode don't modify adsr, it can be choosen by MIDI Wind Controller player for example.
  The musician gives velocity/breath on each note (n1,n2,n3).

3.7.6. Legato playing through InstrumentZone and PresetZone in SoundFont

Soundfont preset are often designed with multiple key ranges and velocity ranges at Instrument Level. That means that when playing a legato passage n1,n2,n3 , as n2,n3 makes use of n1 voices, if n2 or n3 are still in the KeyRange, VelRange of InstrumentZone or PresetZone of the running voices.
If n2,n3 are outsideVelRange and KeyRange, the voices of n1 are forced in release section and new voices are allocated (see notes). So on noteOn the logic respects the keyRange or velrange switching as instructed by the preset in the soundfont.

Notes:
As the voices are necessarily forced in release section, the passage loses its legato caracter at the Instrument Zone transition time.

- This is compensed by a "fast release" for legato mode 0 and a "normal release" for legato mode above 0.
- Note that this can be compensed also at soundfont design level when a Preset Zone have more than one Instruments Zones that overlapps partially.

3.7.7. API set legato mode: **fluid_synth_set_legato_mode(chan,mode)**

FLUIDSYNTH_API
int fluid_synth_set_legato_mode(fluid_synth_t* synth, int chan, int legatomode)

The API function set the legato mode for a channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) to set
- ***legatomode***
  0: RETRIGGER_0 (fast release) 1: RETRIGGER_1 (normal release)
  2: MULTI_RETRIGGER 3: SINGLE_TRIGGER_0 4 : SINGLE_TRIGGER_1

On Output
- FLUID_OK if success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - legatomode is invalid.

Note: The default shell have equivalent command "**setlegatomode**" to set legato mode (see 3.7.9).

3.7.8. API get legato mode : **fluid_synth_get_legato_mode(chan,mode)**

FLUIDSYNTH_API
int fluid_synth_get_legato_model(fluid_synth_t* synth, int chan, int *legatomode)

The API function get the legato mode for a channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) to get
- ***legatomode*** pointer to returned mode.
  0: RETRIGGER_0 (fast release) 1: RETRIGGER_1 (normal release)
  2: MULTI_RETRIGGER 3: SINGLE_TRIGGER_0 4 : SINGLE_TRIGGER_1

On Output
- FLUID_OK if success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - legato is NULL.

Note: The default shell have equivalent command "**getlegatomode**" to display legato mode (see 3.7.10).

3.7.9. command to set legato mode:**setlegatomode**

setlegatomode chan1 Mode1 [chan2 Mode2 ….]

This command uses function API fluid_synth_set_legato_mode() (3.7.7).

3.7.10. command to print legato mode:**legatomode**

legatomode
Print legato mode of all MIDI channels
example

channel: 0, (2)single-trigger_0
channel: 1, (1)multi-retrigger
channel: 2, (0)retrigger_0
channel: 3, (3)single-trigger_1

legatomode chan1 chan2
Print only legato mode of MIDI channel chan1, chan2

This command uses function API fluid_synth_get_legato_mode() (3.7.8).

3.8. Breath mode

This chapter gives details about the presentation given in chapter 3.4.4

3.8.1. API get defaulth breath mode : **fluid_synth_set_breath_mode(chan, breathmode)**

FLUIDSYNTH_API
int fluid_synth_set_breath_mode(fluid_synth_t* synth, int chan, int breathmode)

The API function set the breath mode for a channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) to set
- ***breathmode bits***
  BREATH_POLY default breath modulator poly On/Off
  BREATH_MONO default breath modulator mono On/Off
  BREATH_SYNC breath noteOn/noteOff triggering On/Off

On Output
- FLUID_OK if success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.

Note: The default shell have equivalent command "**setbreathmode**" to set breath mode (see 3.8.3).

3.8.2. API get breath mode : **fluid_synth_get_ breath_mode(chan,breathmode)**

FLUIDSYNTH_API
int fluid_synth_get_ breath_mode(fluid_synth_t* synth, int chan, int *breathmode)

The API function get the breath mode option for a channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) to get
- ***breathmode*** pointer to returned breath mode bits.
  BREATH_POLY default breath modulator poly On/Off

BREATH_MONO default breath modulator mono On/Off
BREATH_SYNC breath noteOn/noteOff triggering On/Off

On Output
- FLUID_OK if success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - breathmode is NULL.

Note: The default shell have equivalent command "**breathmode**" to display breath mode(see 3.8.4).

3.8.3. command to set breath mode:**setbreathmode**

setbreathmode chan1 poly_breath_mod(1/0) mono_breath_mod(1/0) mono_breath_sync(1/0) [..]

Change breath options for channels chan1 and [chan2...]

Parameter 1 is the channel number (i.e 4)
Parameter 2 is the " Breath modulator " enabe/disable for poly mode (i.e disabled)
Parameter 3 is the " Breath modulator " enabe/disable for mono mode (i.e enabled)
Parameter 4 is "breath sync noteOn/Off" enable/disable for mono mode only (i.e enabled)
See presentation in chapter 3.4.4, 3.4.5

This command uses function API fluid_synth_set_default_mode() (3.8.1).

3.8.4. command to print breath mode:**breathmode**

breathmode
Print breath mode of all MIDI channels (poly on/off, mono on/off
example

```
Channel    , poly breath , mono breath , breath sync
channel:  0, off         , off         , off
channel:  1, off         , off         , off
channel:  2, off         , off         , off
.....
```

breathmode chan1 chan2
Print only breath options of MIDI channel chan1, chan2

This command uses function API fluid_synth_get_breath_mode() (3.8.2).

3.9. Portamento mode

Portamento can used on both mode Mono and Poly
Portamento is enabled between from n1 to n2 when Portamento is On on event n1Off and event n2On.
When portamento is Off portamento is disabled.

3.9.1. Portamento 'legato only' in mono mode

Portamento On/Off can be used in mono mode when playing legato.
For example: On two consecutives legato passages n1_1,n1_2,n1_3,.... n2_1,n2_2,n2_3

If portamento is On only during the first passage (n1_1,n1_2,n1_3), there is a portamento
from n1_1 to n1_2 and from n1_2 to n1_3. There is no portamento during the second passage.

If portamento is On during both passage, there is a portamento during both passage, but the first note of each passage (n1_1 and n2_1) are without portamento without the need to release Portamento pedal to get this result.

3.9.2. Portamento 'staccato only' or 'each note' in mono mode

Portamento is possible when playing staccato n1 and n2. The portamento from n1 to n2 occurs even if n1 is released.
In mode 'each note' portamento occurs on each note (staccato or legato) following the first note played.
In mode 'staccato only' portamento occurs only on note played staccato.

PortamentoTime is instructed by CC Portamento time MSB(5d) and LSB(37d)
PortamentoTime is in ms.

3.9.3. Portamento 'legato only' in poly mode

The behavior is the same as mono mode, but note are always played in polyphonic (except when a CC PTC has been received. In this case a portamento with legato mono effect is produced (see 3.4.11))

3.9.4. Portamento 'staccato' in poly mode:

Same as mono (see 3.9.2)

3.9.5. API set portamento mode: **fluid_synth_set_portamento_mode(chan,mode)**

FLUIDSYNTH_API
int fluid_synth_set_portamento_mode(fluid_synth_t* synth, int chan, int portamentomode)

The API function set the portamento mode for a channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) to set
- ***portamentomode*** portamento mode
  0: EACH_NOTE 1: LEGATO_ONLY
  2: STACCATO_ONLY

On Output
- FLUID_OK if success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - portamentomode is invalid.

Note: The default shell have equivalent command "**setportamentomode**" to set portamento mode (see 3.9.7).

3.9.6. API get portamento mode : **fluid_synth_get_portamento_mode(chan,mode)**

FLUIDSYNTH_API
int fluid_synth_get_portamento_model(fluid_synth_t* synth, int chan, int *portamentomode)

The API function get the portamento mode for a channel.

On input
- ***synth*** FluidSynth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) to get
- ***portamentomode*** pointer to returned mode.
  0: EACH_NOTE 1: LEGATO_ONLY
  2: STACCATO_ONLY

On Output
- FLUID_OK if success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - portamentomode is NULL.

Note: The default shell have equivalent command "**portamentomode**" to display portamento mode (see 3.9.8).

3.9.7. command to set portamento mode:**setportamentomode**

setportamentomode chan1 Mode1 [chan2 Mode2 ….]

This command uses function API fluid_synth_set_portamento_mode() (3.9.5).

3.9.8. command to print portamento mode:**portamentomode**

portamentomode

Print portamento mode of all MIDI channels
example

channel: 0, 0-each note
channel: 1, 1-legato only
channel: 2, 2-staccato-only
---
channel: 15, 0-each note

portamentomode chan1 chan2

Print only legato mode of MIDI channel chan1, chan2

This command uses function API fluid_synth_get_portamento_mode() (3.9.6).

3.10. monophonic mode implementation in FluidSynth

3.10.1. ignore MIDI message on MIDI channel disabled

API List

FLUIDSYNTH_API int fluid_synth_noteon(fluid_synth_t* synth, int chan, int key, int vel);
FLUIDSYNTH_API int fluid_synth_noteoff(fluid_synth_t* synth, int chan, int key);
FLUIDSYNTH_API int fluid_synth_cc(fluid_synth_t* synth, int chan, int ctrl, int val);
FLUIDSYNTH_API int fluid_synth_get_cc(fluid_synth_t* synth, int chan, int ctrl, int* pval);
~~FLUIDSYNTH_API int fluid_synth_sysex(fluid_synth_t *synth, const char *data, int len,~~
~~char *response, int *response_len, int *handled, int dryrun);~~
FLUIDSYNTH_API int fluid_synth_pitch_bend(fluid_synth_t* synth, int chan, int val);
FLUIDSYNTH_API int fluid_synth_get_pitch_bend(fluid_synth_t* synth, int chan, int* ppitch_bend);
FLUIDSYNTH_API int fluid_synth_pitch_wheel_sens(fluid_synth_t* synth, int chan, int val);
FLUIDSYNTH_API int fluid_synth_get_pitch_wheel_sens(fluid_synth_t* synth, int chan, int* pval);
FLUIDSYNTH_API int fluid_synth_program_change(fluid_synth_t* synth, int chan, int program);
FLUIDSYNTH_API int fluid_synth_channel_pressure(fluid_synth_t* synth, int chan, int val);

FLUIDSYNTH_API int fluid_synth_bank_select(fluid_synth_t* synth, int chan, unsigned int bank);
handled by fluid_channel_set_sfont_bank_prog()

FLUIDSYNTH_API int fluid_synth_sfont_select(fluid_synth_t* synth, int chan, unsigned int sfont_id);

handled by fluid_channel_set_sfont_bank_prog()

```
FLUIDSYNTH_API int fluid_synth_program_select(fluid_synth_t* synth, int chan, unsigned int sfont_id,
                    unsigned int bank_num, unsigned int preset_num);

FLUIDSYNTH_API int fluid_synth_program_select_by_sfont_name (fluid_synth_t* synth, int chan,
                                const char *sfont_name, unsigned int bank_num,
                                unsigned int preset_num);

FLUIDSYNTH_API int fluid_synth_get_program(fluid_synth_t* synth, int chan, unsigned int* sfont_id,
                    unsigned int* bank_num, unsigned int* preset_num);

FLUIDSYNTH_API int fluid_synth_unset_program (fluid_synth_t *synth, int chan);

FLUIDSYNTH_API int fluid_synth_get_channel_info (fluid_synth_t *synth, int chan,
                                    fluid_synth_channel_info_t *info);

FLUIDSYNTH_API int fluid_synth_program_reset(fluid_synth_t* synth);

FLUIDSYNTH_API int fluid_synth_system_reset(fluid_synth_t* synth);

FLUIDSYNTH_API int fluid_synth_all_notes_off(fluid_synth_t* synth, int chan);
FLUIDSYNTH_API int fluid_synth_all_sounds_off(fluid_synth_t* synth, int chan);

enum fluid_midi_channel_type
{
  CHANNEL_TYPE_MELODIC = 0,
  CHANNEL_TYPE_DRUM = 1
};

int fluid_synth_set_channel_type(fluid_synth_t* synth, int chan, int type);
```

see steps 2.8.4

3.10.2. Insertion point of Poly/mono mode on noteOn et note Off

see fluid_synth_noteon_LOCAL(),fluid_synth_noteoff_LOCAL()

3.10.3. fluid_synth_noteon_LOCAL()

In fluid_chan.h

```
#define fluid_channel_legato(_c)          ((_c)->cc[LEGATO_SWITCH] >= 64)
#define IsChanPlayingMono (_c)  (IsChanMono(_c) || fluid_channel_legato(_c))
```

On noteOn

```
if(IsChanPlayingMono(channel)) /* channel is mono or legato On) */
{ /* monophonic playing */
        fluid_synth_noteon_mono_LOCAL(); /* see fluid_synth_mono.c */
}
else /* channel is Poly with legato off */
{ /* polyphonic playing */
        /* Set the note at first position in monophonic list */
        fluid_channel_set_onenote_monolist();

}
```

see steps 3.11.1

3.10.4. fluid_synth_noteoff_LOCAL()

```
if(IsChanPlayingMono(channel)) /* channel is mono or legato On) */
{ /* monophonic playing */
        fluid_synth_noteoff_mono_LOCAL(); /* see fluid_synth_mono.c */
}
else /* channel is Poly with legato off */
{ /* polyphonic playing */
        /* remove the note from the monophonic list */

}
```

see steps 3.11.1

3.10.5. Using fluidsynth router to simulate a legato pedal by Sustain pedal

In the case of only a sustain pedal is available, you don't need to buy a legato pedal to try legato effect. You can instruct FluidSynth MIDI router to transform a MIDI sustain event to a MIDI legato event. Using fluidsynth application, you need to enter following commands in the shell to instruct the router.

```
# Remove current rules (to remove cc sustain events):
router_clear
# Set the rule to transform CC sustain(64d) to CC legato(68d)
router_begin cc
router_par1 64 64 0 68
router_end
# Set the rules to pass through other messages types (note, prog, pbend, cpress, kpress)
router_begin note
router_end
router_begin prog
router_end
router_begin pbend
router_end
router_begin cpress
router_end
router_begin kpress
router_end
```

3.10.6. monophonic algorithm implementation

Monophonic algorithm (see steps 3.11.3).
Playing staccato noteOn: **fluid_synth_noteon_mono()** (see steps 3.11.4).
Playing noteOff poly ou mono: **fluid_synth_noteoff_polymono()**(see steps 3.11.5).
Playing legato: **fluid_synth_noteon_mono_legato()** (see steps 3.11.6).
Playing noteon legato: **fluid_synth_noteon_mono_legato_retrigger()**(see steps **Erreur! Source du renvoi introuvable.**).
Playing noteon legato: **fluid_synth_noteon_mono_legato_multi_retrigger()** (see steps **Erreur! Source du renvoi introuvable.**).
Playing noteon legato: **fluid_synth_noteon_mono_legato_single_trigger()** (see steps 3.11.7).

3.11. Part 2: Implementations steps in FS.

3.11.1. integrate Polyphonic/monophonic on noteOn and note Off

to do	comments
	fluid_chan.h

done	macros #define fluid_channel_legato(_c) ((_c)->cc[LEGATO_SWITCH] >= 64) ##define IsChanPlayingMono(chan) (IsChanMono(chan) || fluid_channel_legato(chan))

to do	**comments**
	fluid_synth.c.
done	add in fluid_synth_noteon_LOCAL() (3.10.3)
done	add in fluid_synth_noteoff_LOCAL() (3.10.4)
done	declaration extern fluid_synth_noteon_mono_LOCAL(),fluid_synth_noteoff_mono_LOCAL()

to do	**comments**
	new file: fluid_synth_mono.c
done	add fluid_synth_noteon_mono_LOCAL(),fluid_synth_noteoff_mono_LOCAL()
done	Test canal 1 mono, on noteOn/noteOff
done	Test canal 0 poly legato off on noteOn/noteOf
done	Test canal 0 poly , legato on, with legato pedal (3.10.5)

3.11.2. Adding monophonic list

to do	**comments**
	fluid_chan.h
done	In fluid_chan.h, add monophonic list #define maxNotes 10 /* Size of the monophonic list */ struct mononote { unsigned char prev; /* previous note */ unsigned char next; /* next note */ unsigned char note; /* note */ unsigned char vel; /* velocity */ } dans _fluid_channel_t, ajouter struct mononote monolist[maxNotes]; /* monophonic list */ unsigned char iFirst; /* First note index */ unsigned char iLast; /* most recent note index since the most recent add */ unsigned char PrevNote; /**< previous note of the most recent add */ unsigned char nNotes; /* actual number of notes in the list **/ #define LEGATO_PLAYING 0x80 /* b7, 1: means legato playing , 0: means staccato playing */ #define IsChanLegato(chan) (chan->mode & LEGATO_PLAYING) #define SetChanLegato(chan) (chan->mode |= LEGATO_PLAYING) #define ResetChanLegato(chan) (chan->mode &= ~ LEGATO_PLAYING

to do	**comments**
	fluid_chan.c
done	In fluid_chan.c – fluid_channel_init() add list initialization (see 3.6.1)

to do	comments
	fluid_synth_mono.c
done done done	Add functions fluid_channel_add_monolist(), called in fluid_synth_noteon_mono_LOCAL(), test. fluid_channel_search_monolist(), called in fluid_synth_noteoff_mono_LOCAL(), test. fluid_channel_ remove_monolist(), called in fluid_synth_noteoff_mono_LOCAL(), test.
done	void fluid_channel_keep_lastnote_monolist(fluid_channel_t* chan)
done	void fluid_channel_set_onenote_monolist(fluid_channel_t* chan)
done	void fluid_channel_clear_monolist(fluid_channel_t* chan)

to do	comments
	fluid_synth.c
done done	In fluid_synth_cc_LOCAL() , on legato off call fluid_channel_keep_lastnote_monolist(). Test legato On, noteOn, legato Off.
done	In fluid_synth_noteon_LOCAL(), in poly mode, call fluid_channel_set_onenote_monolist(), test.
done	In fluid_synth_noteoff_LOCAL(),in poly mode, call fluid_channel_clear_monolist(), test.

3.11.3. Monophonic algorithm

to do	comments
	fluid_synth_mono.c
	Add algorithm in fluid_synth_noteon_mono_LOCAL(),fluid_synth_noteoff_mono_LOCAL(). Test: on canal 0 polyphonic:
done	// Test1 polyphonic
done	// Test2 monophonique staccato
done	// Test3 monophonique legato
done	// Test4 monophonique legato, legato Off, with one note in the list
done	// Test5 note polyphonique legato with one note monophonic

3.11.4. staccato noteOn: **fluid_synth_noteon_mono()**

to do	comments
	fluid_chan.h, fluid_chan.c
done	In fluid_chan.h, add **key_sustained** in _fluid_channel_t.
done	In fluid_chan.c - fluid_channel_init, initialization key_sustained to – 1;

to do	comments
	fluid_synth_mono.c
done	In **fluid_synth_mono.c,** add function **fluid_synth_noteon_mono()** , call in fluid_synth_noteon_mono_LOCAL().
done	In **fluid_synth.c,** fluid_synth_release_voice_on_same_note_LOCAL() is used by mono algorithm. It need to be public in fluid_synt.c and extern in **fluid_synth_mono.c.**
done	In fluid_synth-fluid_synth_release_voice_on_same_note_LOCAL().to optmize, the function return immediatly when there are no sustained notes.
done	Test 2 (see 3.11.3) to check **fluid_synth_noteon_mono()** is called

3.11.5. noteOff poly or mono: **fluid_synth_noteoff_monopoly ()**

to do	comments
ok	**fluid_voice.h, fluid_voice.c**
done	function fluid_voice_noteoff() is called for poly and mono. It is changed to return True when the note is sustained.

to do	comments
ok	**fluid_synth_mono.c**
done	Add function **fluid_synth_noteoff_monopoly(),** call in fluid_synth_noteoff_mono_LOCAL(). Test 2 (3.11.3)
done	Test 2 (3.11.3) check **fluid_synth_noteoff_monopoly()** is called.

to do	comments
ok	**fluid_synth.c**
done	Update key_sustained, in fluid_synth_damp_voices_by_sustain_LOCAL(), and in fluid_synth_damp_voices_by_sostenuto_LOCAL
done	Test6, note monophonic staccato sustained by sustain
done	call **fluid_synth_noteoff_monopoly()** in fluid_synth_noteoff_LOCAL() to optimize

3.11.6. noteon mono legato: **fluid_synth_noteon_mono_legato ()**

to do	comments
ok	**fluid_synth_mono.c**
done	add function **fluid_synth_noteon_mono_legato()** and call in fluid_synth_noteon_mono_LOCAL(),fluid_synth_noteoff_mono_LOCAL()
done	Test

to do	**comments**
	fluid_rvoice.h, fluid_rvoice.c **fluid_rvoice_event.c**
done	add variable prev_attenuation in fluid_rvoice.h - _fluid_rvoice_dsp_t declaration fluid_rvoice_multi_retrigger_attack() in fluid_rvoice.h
done	add in fluid_rvoice.c - fluid_rvoice_set_attenuation()
done	add function fluid_rvoice_multi_retrigger_attack (fluid_rvoice_t* voice) in fluid_rvoice.c
done	add in fluid_rvoice_event_dispatch() EVENTFUNC_0(fluid_rvoice_multi_retrigger_attack, fluid_rvoice_t*);

to do	comments
	fluid_voice.h, fluid_voice.c
done	add in fluid_voice.c - fluid_update_multi_retrigger_attack()

done	Declaration in fluid_voice.h - fluid_update_multi_retrigger_attack ()

	Mode 1:"muilti-retrigger" , next
to do	**comments**
	fluid_synth_mono.c
done	call fluid_update_multi_retrigger_attack () in fluid_synth_mono.c - fluid_synth_noteon_mono_legato_multi_retrigger()

3.11.7. noteon mono legato: **fluid_synth_noteon_mono_legato (single_trigger())**

Mode 2: single-trigge_0", Mode 3: single-trigge_1",

done done	correction dans fluid_adsr_env.h - fluid_adsr_env_calc() Minor bug: • section count duration (env->count) was 1 less expected. • Whith this patch, section count duration is the right count.

done done	add function: **fluid_rvoice_single_trigger** in fluid_rvoice.c. Declaration in fluid_rvoice.h.
done	add EVENTFUNC_R1(fluid_rvoice_single_trigger, fluid_rvoice_t*) in fluid_rvoice_event.c
done done	add fluid_update_single_trigger() in fluid_voice.c Declaration in fluid_voice.h.
done	call fluid_update_single_trigger() in fluid_synth_mono.c
done done done	enhancement fluid_rvoice_single_trigger() add variable dsp.prev_eff_attenuation in fluid_rvoice.h Initialization to –1 in fluid_rvoice.c - fluid_rvoice_reset()

3.11.8. Portamento

to do	**comments**
	fluid_rvoice.h, fluid_rvoice.c, fluid_voice.h , fluid_voice.c **fluid_rvoice_event.c**
done	Add variable dsp.pitchoffset , dsp.pitchinc in fluid_rvoice.h - _fluid_rvoice_dsp_t declaration void fluid_rvoice_set_portamento() in fluid_rvoice.h
done	init variables dsp.pitchoffset , dsp.pitchinc in fluid_rvoice.c- fluid_rvoice_reset()
done	add function fluid_rvoice_set_portamento () in fluid_rvoice.c
done	adding in fluid_rvoice_event_dispatch() EVENTFUNC_IR(fluid_rvoice_set_portamento, fluid_rvoice_t*);
done done	add function fluid_voice_update_portamento() in fluid_voice.c declaration in fluid_voice.h
done	add portamento in dsp
done done	add function **fluid_voice_calculate_pitch()** in fluid_voice.c. add **fluid_voice_calculate_pitch()** in fluid_voice_calculate_gen_pitch().
done	Test 10: portamento

to do	**comments**
	fluid_chan.h
done done	Add macros #define fluid_channel_portamentotime(_c) #define fluid_channel_portamento()

done	add function fluid_voice_portamento() in fluid_synth_mono.c
done	Integration in fluid_synth_noteon_mono_legato (multi_retrigger)

Portamento for mode 0 retrigger
for mode 1,2,3 (when previous note is finished)

to do	comments
	fluid_synth.h, fluid_synth.c **fluid_voice.c**
done	add fromkey_portamento in fluid_synth.h - struct _fluid_synth_t
done	initialization _ –1 in fluid_synth - new_fluid_synth()
done	add start portamento in fluid_voice.c - fluid_voice_calculate_runtime_synthesis_parameters()
done	triggering in fluid_synth_noteon_mono_legato (retrigger)
done	triggering , • in fluid_synth_noteon_mono_legato (multi_retrigger) • fluid_synth_noteon_mono_legato (single_trigger0) • fluid_synth_noteon_mono_legato (single_trigger1) when previous note is finished.

3.11.9. implementation API legato mode

see 3.7.8

to do	comments
	fluid_chan.h
done	Add ***legatomode*** in struct _fluid_channel_t
done	** acces to channel legato mode ** ** SetChanLegatoMode set the legato mode for a MIDI channel ** #define SetChanLegato(chan,mode) \ (chan->legatomode = mode) ** GetChanLegatoMode get the legato mode for a MIDI channel ** #define GetChanLegatoMode(chan) (chan->legatomode) ** End of macros interface to legato mode variables **

to do	comments
	fluid_chan.c
done	in fluid_chan.c - fluid_channel_init(fluid_channel_t* chan) initialization legato mode.

to do	comments
	synth.h
done	/* API: mono legato mode */ */* Macros interface to mono legato mode variable */* /* n1,n2,n3,.... is a legato passage. n1 is the first note, and n2,n3,n4 are played legato with previous note. n2,n3,..make use of previous voices if any */ enum LegatoMode {

	};
done done	int fluid_synth_set_legato_model(fluid_synth_t* synth, int chan, int legatomode) int fluid_synth_get_legato_model(fluid_synth_t* synth, int chan, int *legatomode)

to do	**comments**
	fluid_synth_polymono.c
done	function **fluid_synth_set_legato_mode()** (3.7.7). Test Ok
done	function **fluid_synth_get_legato_mode()** (3.7.8). Test Ok

3.11.10. Implementation commands legato mode

see presentation 3.7.9

to do	**comments**
	fluid_cmd.c
done	add entry in fluid_commands
	fluid_cmd.h
done	add functions declaration

to do	**comments**
	fluid_synth_polymono.c
done	**legatomode** test Ok
done	**setlegatomode** chan1 Mode1 [chan2 Mode2 ….] test Ok

3.11.11. implementation API: mode Default Breath controller

to do	**comments**
	synth.h
	fluid_chan.h
done	In fluid_chan.h */* Macros interface to breath mode variables */* #define MASK_BREATH_MODE (BREATH_POLY|BREATH_MONO) */* access to default breath infos */* */* SetBreathInfos set the Default breath infos for a MIDI channel */* #define SetBreathInfos(chan,BreathInfos) \ (chan->mode = (chan->mode & ~ MASK_BREATH_MODE) | (BreathInfos & MASK_BREATH_MODE)) #define GetBreathInfos(chan) (chan->mode & MASK_BREATH_MODE) */* GetChanLegatoMode get the legato mode for a MIDI channel */* #define GetChanLegatoMode(chan) (chan->legatomode) */* End of macros interface to legato mode variables */*

done	in synth.h

	/* Interface to default breath state */
	/* bits basic channel infos */
	#define BREATH_POLY 0x10 /* b4, 1: default breath poly On */
	#define BREATH_MONO 0x20 /* b5, 1: default breath mono On */
	/* access to Breath mode bits */
	#define IsPolyDefaultBreath(breath) (breath & BREATH_POLY)
	#define SetPolyDefaultBreath(breath) (breath |= BREATH_POLY)
	#define ResetPolyDefaultBreath(breath) (breath &= ~ BREATH_POLY)
	#define IsMonoDefaultBreath(breath) (breath & BREATH_MONO)
	#define SetMonoDefaultBreath(breath) (breath |= BREATH_MONO)
	#define ResetMonoDefaultBreath(breath) (breath &= ~ BREATH_MONO)
	FLUIDSYNTH_API int fluid_synth_set_breath_mode(fluid_synth_t* synth,
	int chan, int breathmode);
	FLUIDSYNTH_API int fluid_synth_get_breath_mode(fluid_synth_t* synth,
	int chan, int *breathmode);

3.11.12. implementation: Commands Breath mode

to do	comments
	fluid_cmd.c
done	add entry in fluid_commands **breathmode**, **setbreathmode**
	fluid_cmd.h
done	add functions declaration: fluid_handle_setbreathmode(),fluid_handle_setbreathmode()

to do	comments
	fluid_synth_polymono.c
done	**breathmode** test Ok
done	**setbreathmode** chan1 poly_breath(1/0) mono_breath(1/0) [….] test Ok

3.11.13. implementation: mode Default Breath controller

see presentation in 3.4.5

in fluid_synth_alloc_voice()

```
if (!mono && DefaultBreathPoly) || (mono && DefaultBreathMono))
{
        add default modulator: CC Breath To Initial Attenuation.
}
else
{
        add default modulator: Velocity To Initial Attenuation (voir R1).
```

}

	fluid_synth.c
done	Add modulator modulateur fluid_mod_t ***default_breath2att_mod***
done	Initialization of ***default_breath2att_mod*** in fluid_synth_init().
done	add in fluid_synth_alloc_voice()

3.11.14. sleep command

to do	**comments**
	src/bindings/fluid_cmd.c **src/ bindings/fluid_cmd.h**
done	In **src/bindings/fluid_cmd.c** add /* Sleep command, useful to insert a delay between commands */ { "sleep", "general", (fluid_cmd_func_t) fluid_handle_sleep, NULL, "sleep duration sleep duration(in ms)" },
done	**src/bindings/fluid_cmd.h,** add declaration. int fluid_handle_sleep(fluid_cmd_handler_t* handler, int ac, char** av, fluid_ostream_t out);
done	**src/bindings/fluid_cmd.c,** add function fluid_handle_sleep(). int fluid_handle_sleep(fluid_cmd_handler_t* handler, int ac, char** av, fluid_ostream_t out);

3.11.15. implementation: legato trough more zones IZ and PZ

to do	**comments**
	src/sfloader/fluid_defsfont.h **src/sfloader/fluid_defsfont.c**
done	b0) In fluid_defsfont.h- **_fluid_inst_zone_t**, add field **unsigned char flags** add **#define IGNORE_IZ 0x01** **#define IsIgnoreInstZone (iz) (iz->flags & IGNORE_IZ)** **#define SetIgnoreInstZone (iz) (iz->flags | = IGNORE_IZ)** **#define ResetIgnoreInstZone (iz) (iz->flags &= ~IGNORE_IZ)**
done	in **fluid_defsfont.c**, new_fluid_inst_zone() **zone->flags = 0;**
done	b1) in **fluid_inst_import_sfont()**, add parameter ***fluid_preset_zone_t* zonePZ.***
done	b2)In **fluid_preset_zone_import_sfont()** pass parameter ***fluid_preset_zone_t* zonePZ*** to fluid_inst_import_sfont().
done	b3) In **fluid_inst_zone_import_sfont()**, add parameter ***fluid_preset_zone_t* zonePZ.***
done	b4) In **fluid_inst_import_sfont()**, pass ***zonePZ*** to fluid_inst_zone_import_sfont().
done	b5) In **fluid_inst_zone_import_sfont().** for (count = 0, r = sfzone->gen; r != NULL; count++) { sfgen = (SFGen *) r->data; switch (sfgen->id) { case GEN_KEYRANGE: zone->keylo = (int) sfgen->amount.range.lo; zone->keyhi = (int) sfgen->amount.range.hi; break; case GEN_VELRANGE: zone->vello = (int) sfgen->amount.range.lo;

	zone->velhi = (int) sfgen->amount.range.hi; break; default: /* FIXME: some generators have an unsigned word amount value but i don't know which ones */ zone->gen[sfgen->id].val = (fluid_real_t) sfgen->amount.sword; zone->gen[sfgen->id].flags = GEN_SET; break; } r = fluid_list_next(r); } ***/* adjust IZ keyrange to integrate PZ keyrange */*** **if (zonePZ->keylo > zone->keylo) zone->keylo = zonePZ->keylo;** **if (zonePZ->keyhi < zone->keyhi) zone->keyhi = zonePZ->keyhi;** ***/* adjust IZ velrange to integrate PZ velrange */*** **if (zonePZ->vello > zone->vello) zone->vello = zonePZ->vello;** **if (zonePZ->velhi < zone->velhi) zone->velhi = zonePZ->velhi;**
done	Test that code is neutral
	src/sfloader/fluid_defsfont.h **include/fluidsynth/types.h** **include/fluidsynth/synth.h,** **src/synth/fluid_synth.c,** **src/sfloader/fluid_defsfont.c,** **src/sfloader/fluid_ramsfont.c,** **src/synth/fluid_voice.c,** **src/synth/fluid_voice.h**
done	In **src/sfloader/fluid_defsfont.h** – move * FORWARD DECLARATIONS */ typedef struct _fluid_defsfont_t fluid_defsfont_t; typedef struct _fluid_defpreset_t fluid_defpreset_t; typedef struct _fluid_preset_zone_t fluid_preset_zone_t; typedef struct _fluid_inst_t fluid_inst_t; **//typedef struct _fluid_inst_zone_t fluid_inst_zone_t;** in **include/fluidsynth/types.h** typedef struct _fluid_preset_t fluid_preset_t; /**< SoundFont preset */ **typedef struct _fluid_inst_zone_t fluid_inst_zone_t;** /**< Soundfont Instrument Zone */ typedef struct _fluid_sample_t fluid_sample_t; /**< SoundFont sample ***/*** This is necessary to expose fluid_inst_zone_t at API level**, as** fluid_synth_alloc_voice() is a function API.
done	In **include/fluidsynth/synth.h** fluid_synth_alloc_voice(), change parameter ***fluid_sample_t* sample*** by ***fluid_inst_zone_t* inst_zone*** //FLUIDSYNTH_API fluid_voice_t* fluid_synth_alloc_voice(fluid_synth_t* synth, // **fluid_sample_t* sample**, int channum, int key, int vel); FLUIDSYNTH_API fluid_voice_t* fluid_synth_alloc_voice(fluid_synth_t* synth, **fluid_inst_zone_t* inst_zone**, int channum, int key, int vel);
done	In **src/synth/fluid_synth.c** In fluid_synth_alloc_voice(), change parameter ***fluid_sample_t* sample*** to ***fluid_inst_zone_t* inst_zone***

done	In **src/sfloader/fluid_defsfont.c** b7)fluid_defpreset_noteon() passe ***fluid_inst_zone_t inst_zone*** to **fluid_synth_alloc_voice()** // voice = fluid_synth_alloc_voice(synth, **sample**, chan, key, vel); voice = fluid_synth_alloc_voice(synth, **inst_zone**, chan, key, vel);
done	In **src/sfloader/fluid_ramsfont.c** b8) fluid_rampreset_noteon() pass ***fluid_inst_zone_t inst_zone*** to **fluid_synth_alloc_voice()** // voice = fluid_synth_alloc_voice(synth, **sample**, chan, key, vel); voice = fluid_synth_alloc_voice(synth, **inst_zone**, chan, key, vel);
done	In **src/synth/fluid_voice.c** b9) In fluid_voice_init(), change parameter ***fluid_sample_t* sample*** by ***fluid_inst_zone_t *inst_zone*** //fluid_voice_init(fluid_voice_t* voice, **fluid_sample_t* sample**, fluid_voice_init(fluid_voice_t* voice, **fluid_inst_zone_t *inst_zone**,
done	In **src/synth/fluid_voice.h** b10) in fluid_voice_t add field ***fluid_inst_zone_t* inst_zone***
done	In **src/synth/fluid_voice.c** Add #include "fluid_defsfont.h"
done	In **src/synth/fluid_voice.c** In fluid_voice_init(), add **fluid_sample_t* sample = fluid_inst_zone_get_sample(inst_zone)** **voice->inst_zone = inst_zone;**
done	In **src/synth/fluid_synth.c** b12) In **fluid_synth_alloc_voice()** pass **inst_zone** to fluid_voice_init() // if (fluid_voice_init (voice, **sample**, channel, key, vel, if (fluid_voice_init (voice, **inst_zone**, channel, key, vel,
done	b13) Test

to do	**comments**
	src/sfloader/fluid_ramsfont.c **src/sfloader/fluid_defsfont.c**
	In fluid_defsfont.c- fluid_defpreset_noteon() and fluid_ramsfont.c-fluid_rampreset_noteon() ***/* Check if this IZ must be forgotten */*** **ignoreIZ = IsIgnoreInstZone(inst_zone); /* forget is the request to forget this IZ** **ResetIgnoreInsZone(inst_zone); /* Reset the request inside IZ */** /* check if the **note doesn't be forgotten and** falls into the key and velocity range of this instrument zone*/ if **(! ignoreIZ &&** fluid_inst_zone_inside_range(inst_zone, key, vel) && (sample != NULL)) { /* this is a good zone. allocate a new synthesis process and initialize it */ voice = fluid_synth_alloc_voice(synth, **inst_zone**, chan, key, vel); if (voice == NULL) { return FLUID_FAILED; }
done	Test
to do	**comments**
	src/synth/fluid_synth_monoc
done	Add de #include "fluid_defsfont.h"

3.11.16. Implementation of portamento request

to do	comments
to do	**src/synth/fluid_synth_mono.c** **src/synth/fluid_chan.h** **src/synth/fluid_synth.c**
done	In fluid_chan.h add /* Macros interface to monophonic list variables */ /* ChanPrevNote() return the note in iLast entry of the monophonic list */ #define InvalidNote 255 #define IsValidNote(n) (n == InvalidNote) #define ChanPrevNote (chan) (chan->monolist[chan->iLast].note) #define ChanClearPrevNote (chan) (chan->monolist[chan->iLast].note = InvalidNote) /* End of interface to monophonic list variables */
done	In fluid_chan.c - fluid_channel_init() add ChanClearLastNote(chan); /* Mark last note invalid */
done	In fluid_synth.c- fluid_synth_noteoff_LOCAL() if(key == ChanLastNote (channel)) fluid_channel_clear_monolist(channel);
done	In fluid_synth_mono.c, add GetFromKeyPortamentoLegato(channel, defaultFromkey)
done	fluid_synth_mono.c- fluid_synth_noteon_mono_staccato calling GetFromKeyPortamentoLegato() fluid_synth_noteon_mono_legato calling GetFromKeyPortamentoLegato()
done	fluid_synth.c- new_fluid_synth() init de fromkey_portamento to InvalidNote
done done	In fluid_synth_mono.c • Add ValidInvalidLastNoteStaccato(). • In fluid_synth.c-fluid_synth_noteoff_LOCAL() poly – mono call ValidInvalidPrevNoteStaccato()
	Implementation CC PTC in fromkey portamento
done	In fluid_chan.h , add macros: #define clearPortamentoCtrl(_c) ((_c)->cc[PORTAMENTO_CTRL] = InvalidNote) #define PortamentoCtrl(_c) ((_c)->cc[PORTAMENTO_CTRL])
done	In fluid_chan.c , In fluid_channel_init_ctrl() – init PORTAMENTO_CTRL to InvalidNote;
done	In fluid_synth_mono.c – GetFromKeyPortamentoLegato Add CC PTC in GetFromKeyPortamentoLegato()
done done	• In fluid_synth.c, declaration fluid_synth_noteon_mono_legato() extern. • In fluid_synth.c- fluid_synth_noteon_LOCAL call fluid_synth_noteon_mono_legato()

	Add mode legato "Retrigger _1" normal release
	fluidsynth-1.1.6/include/fluidsynth/synth.h enum LegatoMode { /* Release previous note (fast release), start a new note */ RETRIGGER_0, /* mode 0 */ /* Release previous note (normal release), start a new note */ RETRIGGER_1, /* mode 1 */ /* On n2,n3,.. retrigger in attack section using current value and

	shape attack using current dynamic */ MULTI_RETRIGGER, /* mode 2 */ /* On n2,n3,.stay in current value section and shape current section using current dynamic */ SINGLE_TRIGGER_0, /* mode 3 */ /* On n2,n3,.stay in current value section using current dynamic (don't shape adsr) */ SINGLE_TRIGGER_1, /* mode 4 */ LEGATOMODE_NBR };
	fluidsynth-1.1.6/src/synth/fluid_voice.h Update void fluid_update_release(fluid_voice_t* voice, unsigned char flags) **fluidsynth-1.1.6/src/synth/fluid_voice.c** Add parameter flags dans fluid_update_release(fluid_voice_t* voice, unsigned char flags)
	fluidsynth-1.1.6/src/synth/fluid_synth_mono.c In fluid_synth_noteon_mono_legato() **case RETRIGGER_0: /* mode 0 */** **fluid_update_release(voice,0); /* fast release */** **break;** **case RETRIGGER_1: /* mode 0 */** **fluid_update_release(voice,1); /* normal release */** **break;** else { /* tokey note is outside the voice range, so the voice is released */ **fluid_update_release(voice,legatomode);** }
	fluidsynth-1.1.6/src/synth/fluid_synth_polymono.c char * nameLegatomode[LEGATOMODE_NBR]={ "(0)retrigger_0","(1)retrigger_1","(2)multi-retrigger", "(3)single-trigger_0","(4)single-trigger_1"

3.11.17. API portamento mode

to do	**comments**
	\fluidsynth-1.1.6\src\ fluid_chan.h
done	Add ***portamentomode*** in struct _fluid_channel_t
done	*(* acces to channel portamento mode */* *(* SetChanPortamentoMode set the portamento mode for a MIDI channel */* #define SetChanPortamento(chan,mode) \ (chan->portamentomode = mode) *(* GetChanPortamentoMode get the portamento mode for a MIDI channel */* #define GetChanPortamento(chan) (chan->legatomode) *(* End of macros interface to portamento mode variables */*

to do	**comments**
	\fluidsynth-1.1.6\src\synth\fluid_chan.c

done	In fluid_chan.c - fluid_channel_init(fluid_channel_t* chan) init_portamento mode.

to do	comments
	\fluidsynth-1.1.6\include\fluidsynth\synth.h
done	/* API: portamento mode */ */* Macros interface to portamento mode variable */* enum PortamentoMode { /* Portamento on each note (staccato or legato) */ EACH_NOTE, /* mode 0 */ /* Portamento only on legato note */ LEGATO_ONLY, /* mode 1 */ /* Portamento only on staccato note */ STACCATO_ONLY, /* mode 2 */ STACCATOTOMODE_NBR };
done	int fluid_synth_set_portamento_model(fluid_synth_t* synth, int chan, int portamentomode) int fluid_synth_get_ portamento_model(fluid_synth_t* synth, int chan, int * portamentomode)

to do	comments
	fluidsynth-1.1.6\src\synth\fluid_synth_polymono.c
done	function **fluid_synth_set_portamento_mode()**
done	function **fluid_synth_get_ portamento_mode()**

3.11.18. commands_portamento mode

to do	comments
	fluidsynth-1.1.6\src\bindings\fluid_cmd.c
done	add entry in fluid_commands
	fluidsynth-1.1.6\src\bindings\fluid_cmd.h
done	add functions declaration

to do	comments
	fluidsynth-1.1.6\src\synth\fluid_synth_polymono.c
done	**portamentomode** test Ok
done	**setportamentomode** chan1 Mode1 [chan2 Mode2 ….] test Ok

3.11.19. Implementation Breath Sync noteOn/noteOff

see 3.4.5

to do	comments

	fluidsynth-1.1.6\include\fludsynth\synth.h
done	add #define BREATH_SYNC 0x40 /* b6, 1: BreathSyn On */ #define IsBreathSync(breath) (breath & BREATH_SYNC) #define SetBreathSync(breath) (breath |= BREATH_SYNC) #define ResetBreathSync(breath) (breath &= ~ BREATH_SYNC)
	fluidsynth-1.1.6\src\synth\fluid_chan.h
done	1)change #if 1 #define MASK_BREATH_MODE (BREATH_POLY|BREATH_MONO|BREATH_SYNC) #else #define MASK_ BREATH_MODE (BREATH_POLY|BREATH_MONO) #endif #if 1 #define fluid_channel_breath_msb(_c) ((_c)->cc[BREATH_MSB] > 0) #endif 2) add #if 1 #define IsChanBreathSync(chan) IsBreathSync(chan->mode) #define ChanClearPreviousBreath(chan) (chan->previous_cc_breath = 0) #endif #if 1 #define ChanLastVel(chan) (chan->monolist[chan->iLast].vel) #endif 3)In _fluid_channel_t add int previous cc_breath; /**< Previous Breath */
	fluidsynth-1.1.6\src\synth\fluid_chan.c
done	in fluid_channel_init_ctrl() ChanClearPreviousBreath(chan);/* Reset previous breath */
	fluidsynth-1.1.6\src\synth\fluid_synth_polymono.c
done	adding in fluid_handle_breathmode(), fluid_handle_setbreathmode()
	fluidsynth-1.1.6\src\synth\fluid_synth_mono.c
	1)add functions LegatoOnOff(), BreathOnOff()
	fluidsynth-1.1.6\src\synth\fluid_synth.c
done	1)In fluid_synth.c, declaration extern int BreathOnOff(). 2) Remove external declaration fluid_channel_keep_lastnote_monolist(), 3) Add external declaration LegatoOnOff(),BreathOnOff() 4) Adding in fluid_synth_cc_LOCAL() call BreathOnOff(). 5) Adding in fluid_synth_cc_LOCAL 5) case LEGATO_SWITCH: /* Special handling of the monophonic list */ call LegatoOnOff().
	fluidsynth-1.1.6\src\synth\fluid_synth_mono.c
done	Adding in fluid_synth_noteon_mono_LOCAL() int fluid_synth_noteon_mono_LOCAL(fluid_synth_t* synth, int chan, int key, int vel)

	{ int status; fluid_channel_t* channel = synth->channel[chan]; if (!IsChanBreathSync(channel) || fluid_channel_breath_msb(channel)) { } else { /* Add note to the monophonic list */ fluid_channel_add_monolist(channel,(unsigned char)key, (unsigned char)vel); return FLUID_OK; } }
done	Adding in fluid_synth_noteoff_mono_LOCAL ()

3.11.20. Using fluidsynth router to simulate a Breath controller using volume pedal

Using fluidsynth application, you need to enter the following commands in the shell to instruct the router.

Remove current rules (to remove cc sustain events):
router_clear
Set the rule to transform CC volume MSB (7d) to CC breath MSB (2d)
router_begin **cc**
router_par1 7 7 0 2
router_end
Set the rules to pass through other messages types (note, prog, pbend, cpress, kpress)
router_begin note
router_end
router_begin prog
router_end
router_begin pbend
router_end
router_begin cpress
router_end
router_begin kpress
router_end