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 functionalities (see 1).
- Tutorials examples with commands files (see 2.1 and 3.1) (The easy way to learn Poly/mono functionalities).
- Thanks to Ben Gonzales for informations about monophonic accoustic instruments and electronic winds instruments behaviours.

20/10/2017 :
- Correction on typos errors.
- Adding explanations about legato playing through instrument zones and preset zones (see 3.6.3).
- Correction on API documentation (see 2.6.2, 2.6.3).Thanks to Tom Moebert.

28/10/2017 :
- Adding missing explanations about the legato detector (3.4.3).
- Adding example about breathmode (3.4.5).

3/11/2017 :
- Missing complements about using sustain/sostenuto behaviour on mono mode (3.1.8).
- Adding appendices for understanding implementations in FluidSynth (4)

Patch 0004
19/12/2017
- Removing legato modes (0,3,4).
- Keeping legato modes 1 and 2 numbered 0 and 1 respectively.

24/12/2017
- Cleanup Chapter 1.
- Correction on typos errors.

30/12/2017
- Removing basic channel settings (2.2.2).
- Adding comment on Global Channel (3.4.12).
- Changes fluid_synth_set_basic_channel() signature (2.6.3).
- fluid_synth_get_channel_mode() returns basic channel (2.6.4).

18/01/2018

- Adding comments compliant with MIDI spec. for val parameters (2.6.2, 2.6.3).

23/01/2018

Cleanup basic channels API

- Adding a chapter introducing poly/mono basic channels API (2.6.1).
- Removing fluid_basic_channels_infos struct.
- Removing fluid_synth_get_basic_channels() API.
- Clarify fluid_synth_reset_basic_channel(), fluid_synth_set_basic_channel() APIs. (2.6.2)(2.6.3)
- Changes commands resetbasicchannels , setbasicchannels accordling to API changes.
- Replace fluid_synth_get_channel_mode() API by fluid_synth_get_basic_channel() API (2.6.4).

1. Introduction

This document describes poly/mono functionalities for FluidSynth library.
For clarity the document is presented in four chapters.

Chapter 1
- FluidSynth documentation directory content (see 1.1)

Functionalities are in 3 chapters.
Chapter 2
It describes Poly/Mono mode
- Tutorial examples (2.1). Useful examples to learn what is poly/mono mode.
  The easier way to learn quickly Poly/Mono mode.
  Examples are executable directly 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).
- API for channel basic and Poly/Mono mode change(see 2.6).
- shell commands for basic channel and Poly/Mono mode change(see 2.7).

Chapter 3
It describes mainly the monophonic behaviour.
- Tutorial examples (3.1).
  Useful examples to learn what is mono mode behaviour.
  The easier way to learn quickly Mono mode.
  Examples 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 (3.8)
  API, shell commands to handle portamento mode (3.8.6, 3.8.7, 3.8.8, 3.8.9).

- Use of sustain/sostenuto pedal in monophonic mode (3.4.7).
- Use of CC Breath(3.4.5).
  API (3.7.1) and shell commands to handle default breath to Attenuation modulator (3.7.3).
  Option *Breath Sync* to trigger noteOn/noteOff with the breath controller (3.7.3)
- Legato mode playing (3.6).
  legato mode: Retrigger (normal release) and Multi retrigger(3.6.1, 3.6.2)
  API (3.6.4, 3.6.5) and shell commands to handle legato mode (3.6.6, 3.6.7).

- No more limitation (see 3.6.3):
  This patch now accept legato passage through multiples Instruments Zones and Preset Zone.This chapter describes the enhancement to suppress the know unacceptable limitations of previous patches.

- Things not handled
  Sysex message to handle channel basic (2.1.9)

Chapter 4 - Appendices
It describes appendices for understanding implementations in FluidSynth (4).

1.1. Directory content: ./fluidsynt/doc/polymono

- Documentation: FluidPolyMono-0004.pdf.
- Tutorials commands files examples:
  Understanding poly/mono mode: see chapter 2.1
  poly_mono_0.txt.
  poly_mono_1.txt.
  poly_mono_2.txt.
  poly_mono_3.txt.
  poly_mono_4.txt.
  poly_mono_5.txt

  Understanding monophonic mode: see chapter 3.1
  Legato mode: leg_00.txt, leg_01.txt
  Portamento mode: leg_por_00.txt, leg_por_01.txt

2. 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. Tutorial examples – understanding poly/mono mode

This chapter describes useful examples to learn and understand what is poly/mono mode
Example files are executable directly on the FluidSynth console application using the ***source*** command. This is an easy way to learn poly/mono functionalities quickly.

>source command-example-file

The first commande to use at the console is:
>help polymono.

2.1.1. Getting help

The first command to use at the console is:
>help polymono.

basicchannels Prints the list of basic channels
resetbasicchannels [chan1 chan2l..] Resets all or some basic channels
setbasicchannels [chan mode val…] Sets default, adds basic channels
channelsmode [chan1 chan2..] Prints channels mode
legatomode [chan1 chan2..] Prints channels legato mode
setlegatomode chan mode [chan mode..] Sets legato mode
portamentomode [chan1 chan2..] Prints channels portamento mode
setportamentomode chan mode [chan mode..] Sets portamento mode
breathmode [chan1 chan2..] Prints channels breath mode
setbreathmode chan poly(1/0) mono(1/0) breath_sync(1/0) [..] Sets breath mode

These commands, **basicchannels**, **resetbasicchannels**, **setbasicchannels**, **channelsmode** manage basic channels.

2.1.2. What are basic channels ?

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

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

At initialization there is one basic channel: basic channel 0, Omni On Poly (see 2.2).

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

There is only one group starting at basic channel 0. All MIDI channels (0 to 15) inside this group are able to play in polyphonic.

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

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

Assuming you want two 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 the following settings:
- basic channel **5**, mode **poly**, **omni off** (mode **2**), number of channel (dont'care because in this mode only one channel is used see note (a).

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

First, use the command ***resetbasicchannels*** to reset all basic channels:
>***resetbasicchannels***
Warning: no basic channels. All MIDI channels are disabled.
Make use of setbasicchannels to set at least a default basic channel.

Than use the command ***setbasicchannels***:
>**setbasicchannels 5 2** 0 **10 3 1**
or
>setbasicchannels 5 poly_omnioff 0 **10 mono_omnioff 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

resetbasicchannels command resets any previous group and
setbasicchannels allows to set new groups. Now you see 2 groups.

Note a:
- in mode 2, a group is always composed of only one channel regardless of the 3rd parameter.
- in mode 0,1,3 , the 3rd parameter specifies the intended number of channel. When 0, the number is all possible channels from the *basic channel* number to the next basic channel minus 1 (if any) or to MIDI channel count minus 1.

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

2.1.5. 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 to add a new one without modifying actual groups.
Assuming following actual groups (from the previous example):

- 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 3rd new group. 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**
or
>**setbasicchannels 13 mono_omnioff 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
>

Now you see 3 groups.

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

2.1.6. 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 **4** channels in this group.

First use the command ***resetbasicchannels*** to clear the group that need to be changed.
>**resetbasicchannels 5**

Then use command ***setbasicchannels***:
>**setbasicchannels 5 0 4**
or
>**setbasicchannels 5 poly_omnion 4**

Use ***basicchannels*** command to verify your settings
> **basicchannels**
Basic channel: 5, poly omni on (0), nbr: 4
Basic channel: 10, mono omni off(3), nbr: 1
Basic channel: 13, mono omni off(3), nbr: 2
>

Note a: Omni on
- in omni on mode (mode 0 and 1) , a group is always composed of all possible channels from the basic channel number of this group. So the 3rd parameter must not overlap the next basic channel).

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

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

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

The following command changes Group 1 to the following state:

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

Than adds a new group:

- Group :Basic channel: **2**, mono omni on(mode **1**), composed of **3** possible channels in this group (see 2.1.6 Note a)

First use the command ***resetbasicchannels*** to clear the group intended to be changed.
>**resetbasicchannels 5**

Then use command ***setbasicchannels*** to change a group and add a new one
>**setbasicchannels 5 2** 0 **2 1 3**
or
>**setbasicchannels 5 poly_omnioff** 0 **2 mono_omnion 3**

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**

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

Use the command **channelsmode** [chan1 chan2 ….]
It gives details about any channel settings.

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			

>
Note: When a channel is disabled it ignores MIDI NoteOn, noteOff, and CC.

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**

2.1.9. Is there a way to set basic channels via MIDI ?

Sorry, actually in Fluidsynth there no way to set or get basic channels via MIDI cables.

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

The only way to intruct a FluidSynth synthesizer about basic channel is to use shell command or API (2.6).

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

Yes, simply by sending MIDI CC Omni Off, Omni On, Mono On Poly On, messages on this basic channel. (see 2.3 for details)

2.2. "Basic Channel" in FluidSynth

2.2.1. Poly Mono mode numbering convention in FluidSynth

A fluidsynth synthesizer instance may have more than one Basic Channel.
So this instance can work in more than one mode at the same time (see 2.5).

- In the MIDI standard, mode number are 1 based (1 to 4) but ,
- inside FluidSynth mode number are zero based (0 to 3).

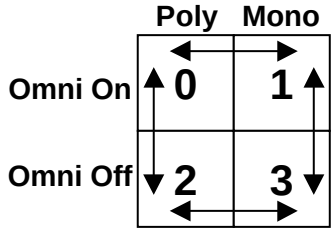


Fig 1. Poly / Mono mode in FluidSynth (Mode number is 0 based).

2.2.2. Basic Channel at synthesizer creation

- At synthesizer creation time (new_fluid_synth()) the synthesizer instance is set with one basic channel: basic channel 0, Omni On Poly On.
  So by default this synthesizer is a polyphonic synthesizer on all MIDI channels.
- Then an application can use the API (2.6) to reset, add or read "Basic channels" informations.

2.2.3. Basic channel commands in Shell

- Default commands shell have new commands to set/print "basic channels informations" (see 2.7).
  So a synthesizer instance can work in "multi mode" (i.e with more than one basic channel) (2.5).

2.3. MIDI Modes messages in FluidSynth

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

Following MIDI CC are handled :
Omni Off (124)**, Omni On** (125)**, Mono On** (126)**, Poly On** (127) only on Basic channel, otherwise messages are ignored.

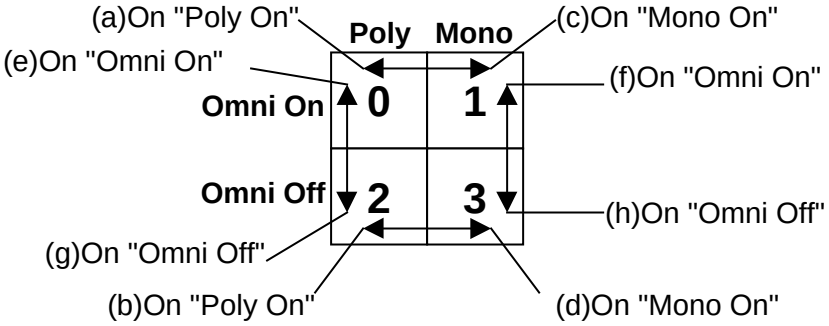


Fig.2: Poly Mono mode change in FluidSynth, usingg MIDI CC messages.

Notes:
For change from mode 0 to mode 3, two CC are need:
- CC Mono On, then CC Omni Off or
- CC Omni Off, then CC Mono On.

For change from mode 1 to mode 2, two CC are need:
- CC Poly On, then CC Omni Off or
- CC Omni Off, then CC Poly On.

2.4. Receiver with one "Basic Channel"

MIDI standard specifies 2 CC messages **Omni Off** (124) and **Omni On** (125) 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 API - 2.6, command - 2.7)).

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

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

Omni On : CC 125 Data=0
Allows listening on all MIDI channels MIDI (0 to 15).
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 others CC messages to set channels in polyphonic or monophonic mode.

Poly On: CC 127 Data=0
Data field is 0.
Sets the MIDI channels in polyphonic.
- If Omni is On, all channels (0 to 16) are listening 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 listening (i.e they are disabled).

Mono On: CC 126 Data=M
Sets the MIDI M channels in monophonic.

- If Omni is On, data field is ignored, all channels (0 à 16) are enabled and monophonic.
- If Omni is Off, data field M is the MIDI channels range (relative to "Basic Channel "), enabled in monophonic. Others channels are disabled.
  Value M to 0 means all channels from "Basic Channel" .

Remark: MIDI standard specify to send Poly On, Mono On messages only 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 enabled in polyphonic.
- Mode 2: Omni On , Mono On Data=0: All channels are enabled in monophonic.
- Mode 3: Omni Off , Poly On Data=0: Basic Channel only is enabled in monophonic.
- Mode 4: Omni Off , Mono On Data=M: Only MIDI channels from "Basic Channel" up to Basic Channel+M-1 are enabled in monophonic.

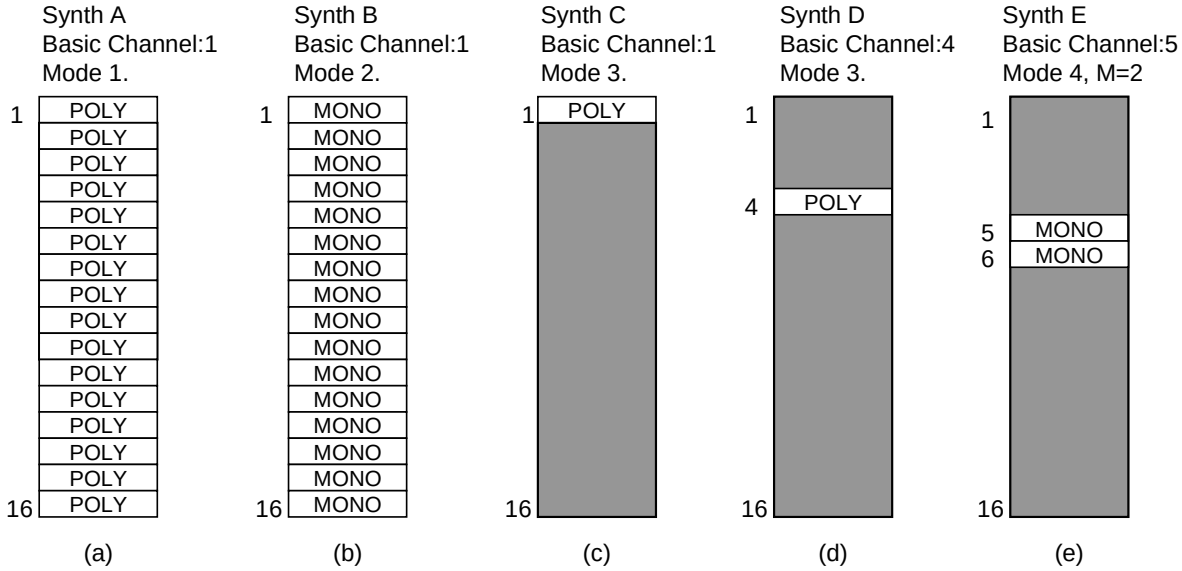


Fig. 1

Notes:

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

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

N2: mode **Omni Off** (mode 3 and 4) allows somes MIDI channels to be disabled . So we can use more than one receiver (C,D,E) in a manner that a MIDI channel can be enabled by only on 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) in mode 3 on "basic channel" 1. MIDI channel 1 is the only one enabled in polyphonic.
- Fig 1.d shows a synthesizer (D) in mode 3 on "basic channel" 4. MIDI channel 4 is the only one enabled in polyphonic.

N3: With **mode 4** it is easy to choose a group of consecutive MIDI channels (M > 1). Think of mode 4 to represent a physical instrument within each channel can play only one note at a time, but all channels can play simultaneously (guitar,bandjo, ..). This mode is suited to strings instruments.

Furthermore, in this mode we can use only one CC to control all the M MIDI channels at the same time (see Global CC 3.4.12).

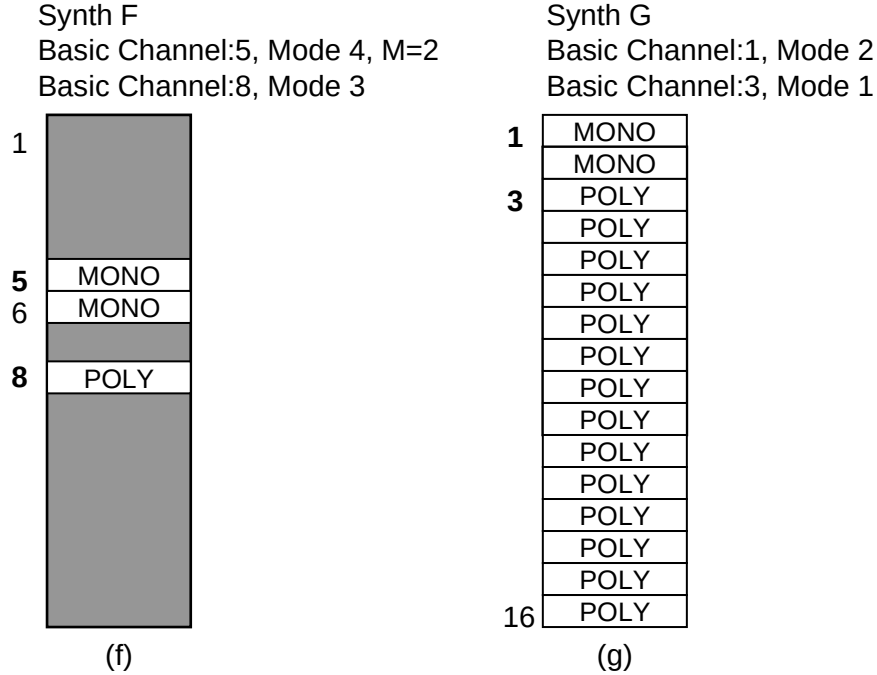
- Fig 1.e show a synthesizer (E) in mode 4 , M=2 on basic channel 5. Only MIDI channels 5,6 are enabled in monophonic.

N4: Note that with only one "Basic Channel" (Fig 1.a to Fig.1 e,) there are no possibility to have some channels polyphonics (mode 1,3) and others channels monophonics (mode 2,4) at the same time.
To overcome this impossibility we need to set more than one "basic channel" (see 2.5).

Note: In the 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 synthesizer (MIDI receiver) can have more than one "basic channel" (MIDI specs(1.0 v 4.2 p7)).



(f) (g)

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 two distinct modes.
- Mode 4 is set on basic channel 5. MIDI channel 5 and 6 are enabled in monophonic.
- Mode 3 is set on basic channel 8. MIDI channel 8 is enabled in polyphonic.
- Others MIDI channels are disabled.

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

When a receiver has 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 to enabled state.

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 enabled in monophonic.
- Mode 1 is set on basic channel 3. MIDI channels 3 to 16 are enabled in polyphonic.

2.6. Poly/Mono mode API in FluidSynth

The Following API are added .These API are used by the new shell commands (2.7).

2.6.1. Purpose of Poly/mono mode API in an application

(1) **fluid_synth_reset_basic_channel()**, **fluid_synth_set_basic_channel()**.
(2) **flluid_synth_get_basic_channel()**.

The couple **fluid_synth_reset_basic_channel()**, **fluid_synth_set_basic_channel()** allows the application to manage the layout of full set of MIDI channels in ***basic channel groups*** in a way fully compliant with the MIDI specs.That is:

(1) **fluid_synth_set_basic_channel()** (2.6.3) allows only to set a ***new basic channel group***, (it doesn't allows to changes an ***existing basic channel group***).
Each time **fluid_synth_set_basic_channel()** specify a ***basic channel*** that overlaps any neighbour existing *basic channel group* the function fails . This occurs when ***basic channel*** parameter overlaps an *existing basic channel group* below, or when ***val*** parameter overlaps an *existing basic channel group* above. Theses overlapping cases can be managed using **fluid_synth_reset_basic_channel()** (2.6.2) that allows to clear (i.e remove) any *existing basic channel group*. (see note).

(2) **fluid_synth_get_basic_channel()** (2.6.4) allows to get information about any MIDI channels ***chan*** regardless it is a *basic channel* or not. These information can be used for general purpose cases like displaying on human interface screen or saving *basic channel group* in permanent storage.
For example the function helps to know:

- if ***chan*** is enabled or disabled. When enabled it means that ***chan*** is member of *a basic channel group*.
- if ***chan*** is equal to the returned ***basic_chan_out*** parameter it means that *chan* is the ***first channel*** of a *basic channel group* and the returned ***val_out*** parameter is the numbers of channels in this *group*.

To get information about all MIDI channels the application need to itterate until the count of MIDI channels returned by **fluid_synth_count_midi_channels()** API.

Note:
Note that MIDI specifications allows to change an existing basic channel using MIDI CCs ***omni off***, ***omni on***, ***mono***, ***poly*** (see 2.3) send by MIDI cable or using **fluid_synth_cc()** API. These CC have to be send on a basic channel already set by advance (using **fluid_synth_set_basic_channel()**).

2.6.2. Reset basic channels: **fluid_synth_reset_basic_channel()**

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

Resets a basic channel group designed by basicchan.

On input

- ***synth*** the synthesizer instance.
- ***chan*** the basic channel of the group to reset. -1 means reset all basic channels.

Note: Be aware than when a synth instance has no basic channel, all channels are disabled.
In the intend to get some MIDI channels enabled, the application have to set at least one basic channel using fluid_synth_set_basic_channel API (2.6.3).

Note By default (i.e. on creation after new_fluid_synth() and after fluid_synth_system_reset()) a synth instance has one basic channel at channel 0 in mode FLUID_CHANNEL_MODE_OMNION_POLY.
All other channels belong to this basic channel group.

On Output

- FLUID_OK on success
- FLUID_FAILED,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - chan isn't a basic channel.

2.6.3. Set basic channel: **fluid_synth_set_basic_channel()**

FLUIDSYNTH_API
int fluid_synth_set_basic_channel(fluid_synth_t* synth, int chan, int mode, int val);

Sets a new basic channel group only. The function doesn't allow to change an existing basic channel.

The function fails if any channels overlaps existing basic channel groups. To make room if necessary, neighbourg basic channel groups can be cleared using **fluid_synth_reset_basic_channel()** API (2.6.2).

On input
- ***synth*** the synthesizer instance.
- ***chan*** the basic Channel number (0 to MIDI channel count-1).
- ***mode*** the MIDI mode to use for chan (0 to 3).
  FLUID_CHANNEL_MODE_OMNION_POLY
  FLUID_CHANNEL_MODE_OMNION_MONO
  FLUID_CHANNEL_MODE_OMNIOFF_POLY
  FLUID_CHANNEL_MODE_OMNIOFF_MONO
- ***val*** number of channels in the group (for mode 0,1,3 only).

Note:
- ***val*** is relevant only for mode poly omnion (0), mono omnion (1), mono omni off (3).
  A value of 0 means all possible channels from ***chan*** to the next basic channel minus 1 (if any) or to MIDI channel count minus 1.
- ***val*** is ignored for mode poly omnioff (2) as this mode implies a group of only one channel.

On Output
- FLUID_OK on success
- FLUID_FAIL,
  - synth is NULL.
  - chan is outside MIDI channel count.
  - mode is invalid.
  - val has a number of channels overlapping another basic channel or been above MIDI channel count.

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

2.6.4. Get channel mode: **fluid_synth_get_basic_channel()**

FLUIDSYNTH_API
int fluid_synth_get_basic_channel(fluid_synth_t* synth, int chan,
                                  int *basic_chan_out,
                                  int *mode_out,
                                  int *val_out)

Returns poly mono mode informations about any MIDI channel.

On input
- ***synth*** the synthesizer instance.
- ***chan*** MIDI Basic channel number (0 to MIDI channel count - 1).

- ***basic_chan_out***: Buffer to store the basic channel ***chan*** belongs to (FLUID_FAILED if chan is disabled).
- ***mode_out*** Buffer to store the mode of ***chan***. (FLUID_FAILED if chan is disabled).
  -*mode* (see enum fluid_channel_modes) of chan
  FLUID_CHANNEL_MODE_OMNION_POLY
  FLUID_CHANNEL_MODE_OMNION_MONO
  FLUID_CHANNEL_MODE_OMNIOFF_POLY
  FLUID_CHANNEL_MODE_OMNIOFF_MONO
- ***val_out*** Buffer to store the total number of channel in this basic channel group (FLUID_FAILED if chan is disabled).

Note : if any of ***basic_chan_out***, ***mode_out***, ***val_out*** pointer is NULL the corresponding information isn't returned.

On Output
- FLUID_OK on success
- FLUID_FAILED
  -synth is NULL.
  -chan is outside MIDI channel count.

Note: The default shell has an 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

Prints the list of all MIDI basic channels informations
example:..

```
Basic channel:  0, poly omni on (0), nbr:  3
Basic channel:  3, poly omni off(2), nbr:  1
Basic channel:  8, mono omni off(3), nbr:  2
Basic channel: 13, mono omni on (1), nbr:  3
```

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

resetbasicchannels

With no parameters the command resets all basic channels.
Note: Be aware than when a synthesizer instance has no basic channels, all channels are disabled. In the intend to get some MIDI channels enabled, use the command setbasicchannels.

resetbasicchannels chan1 [chan2 . . .]
Resets basic channel group chan1, basic channel group chan2 . . .

This command uses function API fluid_synth_reset_basic_channels() (2.6.2).

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

setbasicchannels

With no parameters the commands set one channel basic at basicchan 0 in Omni On Poly (i.e all the MIDI channels are polyphonic).

setbasicchannels chan1 mode1 nbr1 [chan2 mode2 nbr2] ... ...

Adds basic channel 1 and 2

The command fails if any channels overlaps any existing basic channel groups. To make room if necessary, existing basic channel groups can be cleared using **resetbasicchannels** command (2.7.2).

Mode can be a numeric value or a name:
- numeric: 0 to 3 or
- name: poly_omnion , mono_omnion, poly_omnioff, mono_omnioff.

This command uses API fluid_synth_set_basic_channel() (2.6.3).

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

channelsmode

Prints 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

Prints only channel mode off MIDI channels chan1, chan2

This command uses API fluid_synth_get_basic_channel () (2.6.4).

3. Polyphonic/monophonic behaviour inside Fluidsynth

This chapter describes the behaviour of a monophonic instrument (like MIDI Wind Controller) when played by a musician (3.2).
It describes also the behaviour 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 MIDI controller (transmitter) connected on its input. This is how FluidSynth behaves.

3.1. Tutorial examples- understanding monophonic behaviour

This chapter describes useful examples to learn and understand what is monophonic behaviour. Example files are executable directly 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 details.)
- 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 by deperessing the 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 releases the legato pedal, the channel restores in the state prior legato On.
In others words legato On/Off is mainly useful 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 sends CC legato On/Off at the appropriate moment.

In summary when a channel receives CC legato On/Off it will be able to reacts monophonically regardless its actual poly/mono mode .
Legato On/Off is a real time performance CC to allowing polyphonic instrument to be played like monophonic instrument.
This doesn't mean that legato On/Off supersedes 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 state.

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 between n1,n2, is articulated on n2On. There are 2 legato modes. When a channel is set in a mode, articulations betwween note transition always make use of this mode.

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

Examples

legato mode 0: **source leg_00.txt**
legato mode 1: **source leg_01.txt (see remark)**

Remark:

- Dependant of the switching preset zones, controlled both by velocity range and key range, legato mode (1) can use temporarily the same articulation than legato mode 0 (see 3.6.3 for details). 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" but "key range switching" are possible.
- If you want to change the preset you can edit the file leg_xx.txt and modify the **prog** command.

See 3.6 for détails about legato mode.

3.1.5. What are breath mode ?

Breath mode allows a keyboard player to control dynamic using 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 attention 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.

Attention:
- 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.

Also, breath mode allows the musician to synchonize noteOn/noteOff using the breath controller.
(see 3.4.4 for details.)

3.1.6. What is portamento ?

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

Examples

legato mode 0, with portamento: **source leg_port_01.txt**
legato mode 1, with portamento: **source leg_ port_01.txt**

Remark:
- Dependant of the switching preset zones, controled both by "velocity range and key range" , legato mode 1 can use temporarily the same articulation than legato mode 0 (see 3.6.3 for details). 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 examples velocity is the same for all the notes, so there is no "velocity range" switching" but "key range switching" are possible.
- If you want to change the preset you can edit the file leg_xx.txt and modify the **prog** command.

See 3.4.9 for details.

3.1.7. What are portamento mode ?

With portamento mode the musician choose the situation in wich portamento occurs. This situation (i.e this mode) is set with portamento mode commands (see 3.8 for details).

3.1.8. What about sustained note in monophonic mode ?

When in mono mode, note can be sustained (by sustain or sostenuto pedal) as in poly mode.
- While a note is sustained, playing staccato noteon/noteoff always release previous sustained note and than sustain the new note if sustain or sostenuto pedal is still depressed. The result sounds like in legato mode 0 (normal release).
- Playing legato while sustain (or sustenuto) is depressed will sustain only the last note of the legato passage.

3.2. monophonic MIDI Wind Controller instrument

This chapter describes MIDI Wind Controller behaviour from informations supplied by Louis B.
- see starting thread [Fluid-dev] Help about MIDI Wind Controller behaviour 30 Apr 2015.

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

Also many thanks to Ben Gonzales for informations about monophonic accoustic instruments and electronic winds instruments behaviours.

3.2.1. Basic behaviour

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 starts when blowing starts. A MIDI noteOn is sended with a velocity value coming from Breath value. Notes value depends of the depressed key.
  While breath continues, 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}, (see R2).
  The noteOn velocity is the current breath value.
  Legato passage can be in ascending order {noteOn 2, noteOff n1},{noteOn n3, noteOff n2} or descending order {noteOn 2, noteOff n3} (see R3).
- Playing stops when the musician stops blowing; a MIDI noteOff is sended with the note pitch egal to the previous noteOn.
- Between starting end ending blowing , if the musician plays and release 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 noteOff must be send for each noteOn sended.

We remark that to reproduce a legato passage, typical synthesizer behaviour 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 musician release a key. For example if n1 than n2 key are depressed we ear note n2. Then when n2 is released we ear note n1.

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

3.2.2. Playing staccato

To start and stop a note n1, the musician starts and stops blowing.

When musician blows in the 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 the note typically comes from MSB value of 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 musician continues to blow at n2 key time while n1 was still depressed. This wasn't not the case at n1 time.

R2: The MIDI Wind Controller detects the beginning 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 (a standard MIDI behaviour).
The synthesizer can ignore this message.

The musician continues 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 doesn't send unnecessary cc legato On.
When receiving n3, the synthesizer remembers of a running legato state, so it reacts legato n2,n3 (same as for R2 above).
Remark R3 above is relevant.

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

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

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 below):

- 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 , as key n1 and n2 are still deppressed, the MIDI Wind Controller sends a noteOn n2 that will 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 maintaining and releasing key n2:

- daten2_off: noteOn n1, vel = current data breathMSB
- daten2_off: noteOff n2, vel=0 (see R3 above)

When the musician releases breath a noteOff is sended (R4 below) eventualy followed by legato Off (R5 below) 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: The synthesizer must play this event, it doesn't ignore it as it done in R3.
R5: The MIDI Wind Controller detects a legato ending because it knows that the played note was the last depressed note.

3.3. Polyphonic controller (Keyboard)

3.3.1. basic behaviour

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

Most polyphonic MIDI controller doesn'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, most polyphonic MIDI controller doesn't send MIDI legato On/Off automatically (has should does MIDI Wind Controller see 3.2).

3.4. Synthesizer monophonic behaviour

This chapter describes the synthesizer behaviour 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).
- Monophonic 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 regardless the playing manner (staccato or legato). Chapter 3.4.2 gives the algorithm when using a polyphonic controller on input.

3.4.2. polyphonic controller (keyboard) to monophonic channel.

When playing staccato on keyboard, synthesizer behaviour is the same regardless the channel mode (Poly/Mono).
When playing legato, it is different. The synthesizer needs to remember of the notes belonging to a legato passage from the 1st note. This memory is necessary to detect a legato passage and to allow a reverse order playing legato (see explanations 3.4.3).

3.4.3. Legato playing detector – the monophonic list

The list allows an easy automatic detection of a legato passage when it is played on a MIDI keyboard input device.
It is useful also when the input device is an ewi (electronic wind instrument) or evi (electronic valve instrument) and these instruments ar unable (like many polyphonic MIDI controllers) to send MIDI CC legato on/off.

The list memorizes the notes in playing order.

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

The features are:

- 1) It is always possible to play an infinite legato passage in direct order (n1_On,n2_On,n3_On,....).
- 2) Playing legato in the reverse order (n10_Off, n9_Off,,...) helps in fast trills playing as the list memorizes 10 most recent notes.
- 3) Playing an infinite lagato passage in ascendant or descendant order, without playing trills is always possible using the usual way like this:
  - First we begin with an ascendant passage: n1On, (n2On,n1Off), (n3On,n2Off) , (n4On,n3Off), then
  - we continue with a descendant passage: (n3On,n4off), (n2On,n3off), (n1On,n2off), n1Off...and so on

Each MIDI channel have a monophonic list.

The following examples shows the purpose of the monophonic list.

Exemple 1 :Imagine a flutist (playing a flute with holes) who wants to play the whole passage in legato playing:

n1,n2,n3,n2,n1

The whole passage is n1,n2,n3 (in direct order) than n2,n1 (in reverse order).

The flutist needs to do:

First playing n1,n2,n3 in direct order:

- Press key n1, No legato detection, n1 is started normally . (1st note the legato passage).
- Press key n2, legato detection (n1,n2) , so n2 is played (using voices of n1).
- Press key n3, legato detection (n2,n3), so n3 is played (using voices of n2).

Then playing n2,n1 in reverse order

- Depress n3, legato detection (n3,n2), so n2 is played (because key n2 was still depressed) (using voices of n3).
- Depress n2, legato detection (n2,n1), so n1 is played (because key n1 was still depressed) (using voices of n2).
- Then depress n1, no legato, n1 is stopped. (end of the legato passage).

Exemple 2: Now imagine the flutist wants to play a trill beetwen n1 and n2. The flutist needs to do: First playing n1On, than n2On than depress /release n2 more times, this produces a trill.

These examples showed :

- how the monophonic list helps to detect a legato between pairs of notes.
- how the legato detector can trigger a note during the reverse order using the note value and velocity value memorized by the list at noteOn time.

On noteOn, the note are added in the monophonic list that works as a circular buffer.
On noteOff, any corresponding note are removed from the list. So when playing in the reverse order the musician have the possibility to ignore some notes.
Example
lets n1,n2,n3,n4 to be played legato in the direct order.
Than the musician wants to continue in the reverse order without n2, so simply he needs to release n2 before releasing n3.

With a monophonic list size of 10 notes, we have

- Low memory cost (30 bytes per channel).
- Fast insertion/removing (at noteOn/noteOff).

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 musician plays 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 soundfont preset designer to add CC Breath support. If a CC "Breath to attenuation" modulator is present in the soundfont, Fluidsynth synthesizer will play it. This is the right way as this 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 this cases:

- there is no CC Breath modulator inside the Soundfont or
- the keyboardist want to control dynamic with only a breath controller (but not with both key velocity + breath controller).

The following 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 (see figure below).

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**

*# Choosing MIDI channel 2 in **Poly** mode (for example Poly Omni Off: 2)*
setbasicchannels 2 2 0
*# Choosing **BreathToAtt** on MIDI Channel 2 when it will be in **Mono** state*
 # (The 3rd parameter must be set to 1).
SetBreathMode 2 0 1

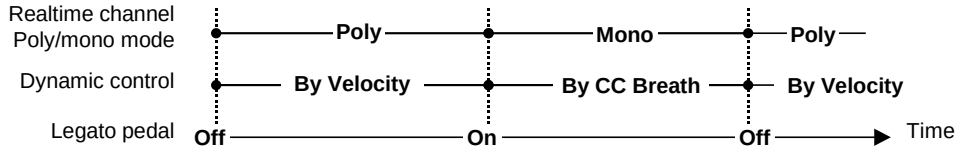


Fig:Realtime source control of dynamic (velocity or CC breath) on a polyphonic MIDI channel.
•When legato pedal is Off, dynamic is controlled by velocity (coming from the keyboard).
•When legato pedal is On, dynamic is controlled by the breath (coming from CC Breath controller).

Triggering notes with the breath controller

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

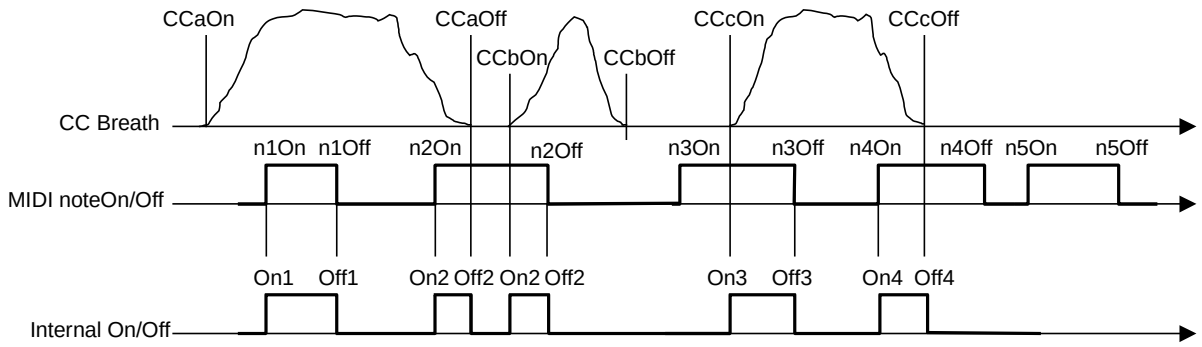
It allows the breath controller (MSB) to trigger noteoff/noteon on the running note. If you are a wind player and you want to play on a keyboard, you would appreciate similar behaviours than playing on electronic wind instruments. So this option is only efficient if the MIDI input device is a keyboard.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

```
setbreathmode 4  0  1  1
```

3rd Parameter is 1 to enable " Breath modulator " for mono mode.
4th 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 musician playing wind instrument.

- 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.
  - Note this option allows to play staccato using a breath controller.



The figure illustrates the "Breath sync noteOn/noteOff" option behaviour.

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

We remark that the behaviour 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 the type of MIDI controller connected to its input.

3.4.7. Using Sustain / Sostenuto in monophonic mode

A note on a monophonic channel be hold by Sustain or Sostenuto, the same way as for a 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/Polyphonic mode
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 / Poly
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 3.8 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 in mono mode.

That means that a mono channel is able to play legato if the musician plays legato.
A musician plays a legato passage n1,n2 when he plays n2 without releasing n1.

Remarks:

- In Fluidsynth, on a poly channel, if note n1 is keeped depressed, then legato pedal is depressed, when n2 is played On, a legato passage begins 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 intrument. Wind players using a keyboard would appreciate identic behaviour.
- When a channel is set in mono mode (by basic channels commands(2.2), or CC poly/mono (2.3)) , it ignores 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. In FluidSynth this behaviour works 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 example while the musician hold C major chord (C,E,G) if CC PTC with value G is received, when the musician plays 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 there is a basic channel in mode 3 (omni off mono), it is possible to send only one CC for all MIDI Channel that belong to this group. The CC is called a global CC.

- To be accepted as global, the CC must be send on a channel one below the basic channel and that channel must be disabled (i.e it must not belong to another group otherwise the CC is considered as a normal CC sent only to one channel).

In Fluidsynth, it is always possible to have more than one basic channel with the use of CC global in perspective. To get this result, the MIDI channel one below the basic channel number must be disabled (see 2.5).

Examples: let a synthesizer with 16 MIDI channels

- Let a basic channel 7 in mode 3 (Omni Off – Mono). If MIDI channel 6 is disabled it could be used as CC global for all channels belonging to basic channel 7.
- Let a basic channel 0 in mode 3. If MIDI channel 15 is disabled it could be used as CC global for all channels belonging to basic channel 0.

3.5. Polyphonic Channel behaviour

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

3.6. 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 which have velocity or a preset with true adsr volume articulations (i.e with attack decay and sustain not at the same level). These modes have little or no effect on "flat" adsr enveloppe.

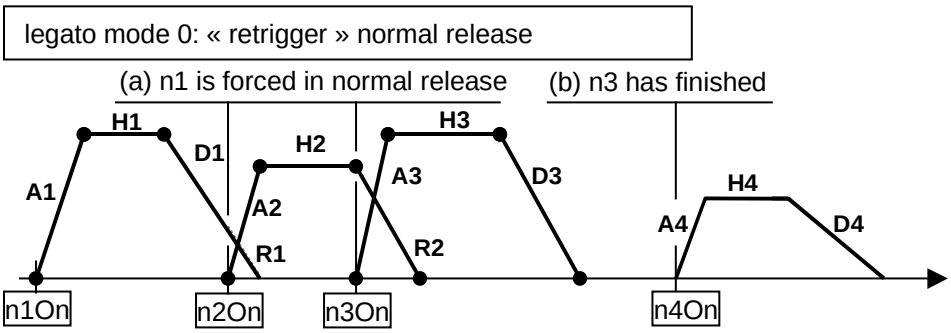
These modes articulate the notes follwing the 1st note differently.

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 articulation used.

There are 2 modes numbered 0 to 1. 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 (3.6.4) or commands (3.6.6).

3.6.1. Mode 0: "retrigger" (normal release)

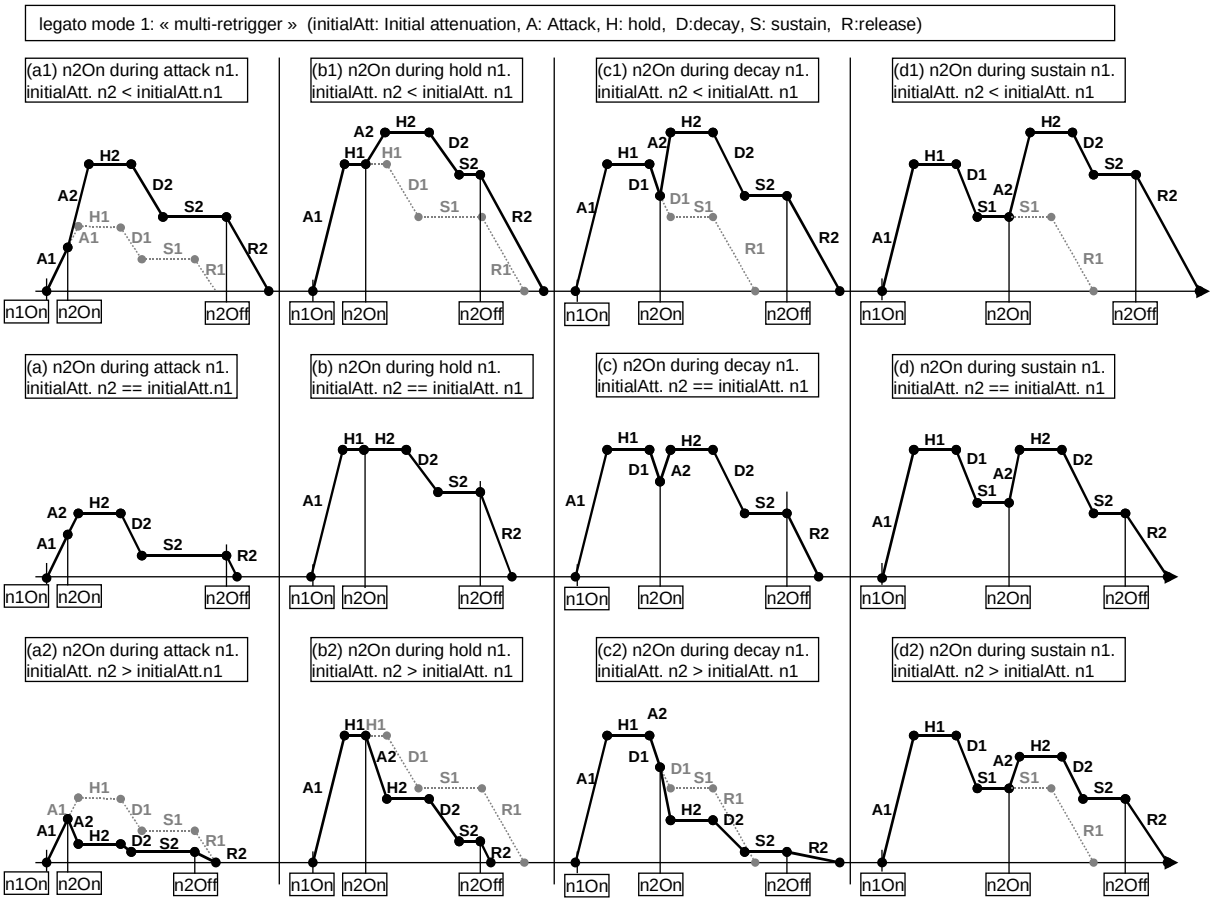


- If release is not too fast there is a cross fading between release of previous note and attack of the current note.

Remark: for example.
This mode is adequat for playing fast "détaché" trills.
If the musician don't want playing trill he just needs to release previous note before releasing current note .

Also, this mode is convenient for percussive instrument (ADSR with a decay).

3.6.2. Mode 1: "multi-retrigger"



With 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 can retriggered during attack section (i.e retrigered more than one time for example if the attack isn' to fast and the following notes (n2,n3) come sooner after n1.

Depending on the envelope shape, this mode is more legato than mode 0.

Note: Dependant of the switching preset zones, controlled both by velocity range and key range , legato mode (1) can use temporarily the same articulation than legato mode 0 (see 3.6.3 for details).

3.6.3. Legato playing through Instrument Zone and Preset Zone in SoundFont

This patch now accepts legato passage through multiples Instruments Zones and Preset Zone.This chapter describes the enhancement to suppress the know (and unacceptable) limitations of previous patches.(see appendices 4.7)

3.6.4. 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 sets the legato mode of a MIDI channel.

On input
- ***synth*** the synth instance.
- ***chan*** MIDI channel number (0 to MIDI channel count - 1) .
- ***legatomode***
  0: FLUID_CHANNEL_LEGATO_MODE_RETRIGGER.
  1: FLUID_CHANNEL_LEGATO_MODE_MULTI_RETRIGGER.
-

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

Note: The default shell has an equivalent command "**setlegatomode**" to set legato mode (see 3.6.6).

3.6.5. 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 gets the legato mode from a MIDI channel.

On input
- ***synth*** the synth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1)
- ***legatomode*** pointer to returned mode.
  0: FLUID_CHANNEL_LEGATO_MODE_RETRIGGER.
  1: FLUID_CHANNEL_LEGATO_MODE_MULTI_RETRIGGER.

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

Note: The default shell has an equivalent command "**getlegatomode**" to display legato mode (see 3.6.7).

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

setlegatomode chan0 mode1 [chan1 mode0] .. ..

This command uses API fluid_synth_set_legato_mode() (3.6.4).

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

legatomode
Prints legato mode of all MIDI channels
example

channel: 0, (1)multi-retrigger
channel: 1, (0)retrigger
channel: 2, (1)multi-retrigger

legatomode chan1 chan2
Prints only legato mode of MIDI channels chan1, chan2

This command uses API fluid_synth_get_legato_mode() (3.6.5).

3.7. Breath mode

This chapter gives details about the presentation given in chapter 3.4.4

3.7.1. 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 sets the breath mode of a MIDI channel.

On input
- ***synth*** the synth instance.
- ***chan*** MIDI channel number (0 to MIDI channel count - 1).
- ***breathmode bits***
  FLUID_CHANNEL_BREATH_POLY default breath modulator poly On/Off
  FLUID_CHANNEL_BREATH_MONO default breath modulator mono On/Off
  FLUID_CHANNEL_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 has an equivalent command "**setbreathmode**" to set breath mode (see 3.7.3).

3.7.2. 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 gets the breath mode option of a MIDI channel.

On input
- ***synth*** the synth instance.

- ***chan*** MIDI channel number (0 to MIDI channel count - 1)
- ***breathmode*** pointer to returned breath mode bits.
  FLUID_CHANNEL_BREATH_POLY default breath modulator poly On/Off
  FLUID_CHANNEL_BREATH_MONO default breath modulator mono On/Off
  FLUID_CHANNEL_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 has an equivalent command "**breathmode**" to display breath mode(see 3.7.4).

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

setbreathmode chan1 poly_breath_mode(1/0) mono_breath_mode(1/0) mono_breath_sync(1/0) [..]

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

Example: setbreathmode 4 0 1 1

Parameter 1 is the channel number (i.e 4)
Parameter 2 is the " Breath modulator " enable/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() (3.7.1).

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

breathmode
Prints breath mode of all MIDI channels

poly breath on/off, mono breath on/off, breath sync 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
Prints only breath options of MIDI channels chan1, chan2

This command uses API fluid_synth_get_breath_mode() (3.7.2).

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

Using fluidsynth application console, 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

3.8. Portamento mode

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

3.8.1. 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 passages, 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 the Portamento pedal to get this result.

3.8.2. Portamento modes **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 following the first note played (staccato or legato).
- In mode **staccato only** portamento occurs only on note played staccato.

3.8.3. Portamento **legato only** in poly mode

The behaviour is the same as in mono mode, but notes 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))

3.8.4. Portamento **staccato** in poly mode:

Same as mono (see 3.8.2)

3.8.5. Portamento time: **CC MSB(5d) and LSB(37d)**

PortamentoTime is instructed by CC Portamento time MSB(5d) and LSB(37d)
PortamentoTime (ms) = 128 x MSB + LSB

MSB,MSB value are 0 to 127 each.
Maximum time: 128 x 127 + 127 = 16,383 second

The resolution of CC 5 is 128 ms
CC 5: 1, time = 128 ms

CC 5: 2, time = 256 ms
CC 5: 3, time = 384 ms
CC 5: 4, time = 512 ms

To get smaller time (< 128 ms) we need to set CC 5 at 0 (which is the default value at fluidsynth initialization), and use only CC 37d

Notes
- To enable portamento CC Portamento 65d must be on
- If portamento time is 0 the pitch sweep is not perceptible.(At fluidsynth initialization: portamento time is 0).
- If pitch bend is send, it is actually understood by FluidSynth as pitch bend (i.e adding pitch shift .This a default behaviour). That means, that Pitch bend can be used during a portamento effect.

3.8.6. 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 sets the portamento mode to a MIDI channel.

On input
- ***synth*** the synth instance.
- ***chan*** MIDI channel number (0 to MIDI channel count - 1).
- ***portamentomode*** portamento mode
  0: FLUID_CHANNEL_PORTAMENTO_MODE_EACH_NOTE.
  1: FLUID_CHANNEL_PORTAMENTO_MODE_LEGATO_ONLY.
  2: FLUID_CHANNEL_PORTAMENTO_MODE_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 has an equivalent command "**setportamentomode**" to set portamento mode (see 3.8.8).

3.8.7. 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 gets the portamento mode of a MIDI channel.

On input
- ***synth*** the synth instance
- ***chan*** MIDI channel number (0 to MIDI channel count - 1)
- ***portamentomode*** pointer to returned mode.
  0: FLUID_CHANNEL_PORTAMENTO_MODE_EACH_NOTE.
  1: FLUID_CHANNEL_PORTAMENTO_MODE_LEGATO_ONLY.
  2: FLUID_CHANNEL_PORTAMENTO_MODE_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 has an equivalent command "**portamentomode**" to display portamento mode (see 3.8.9).

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

setportamentomode chan1 mode1 [chan2 mode2 ….]

Changes portamento mode for channels chan1 and [chan2]

This command uses API fluid_synth_set_portamento_mode() (3.8.6).

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

portamentomode

Prints 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

Prints only portamento mode of MIDI channel chan1, chan2

This command uses API fluid_synth_get_portamento_mode() (3.8.7).

4. Part 4: Appendices for understanding implementations in FluidSynth.

This chapter is useful for developer or reviewer . It helps to localize the relations between the polyphonic and monophonic functionalities inside FluidSynth sources files.

Note that finding monophonic behaviour a bit difficult to understand is a normal situation because it is due to the normal behaviour of monophonic instrument particularly when this instrument is played in a legato manner.

If you are not familiar with monophonic instrument behaviour chapter 3.2 should help you to get more confident.

4.1. Polyphonic and monophonic functions in FluidSynth

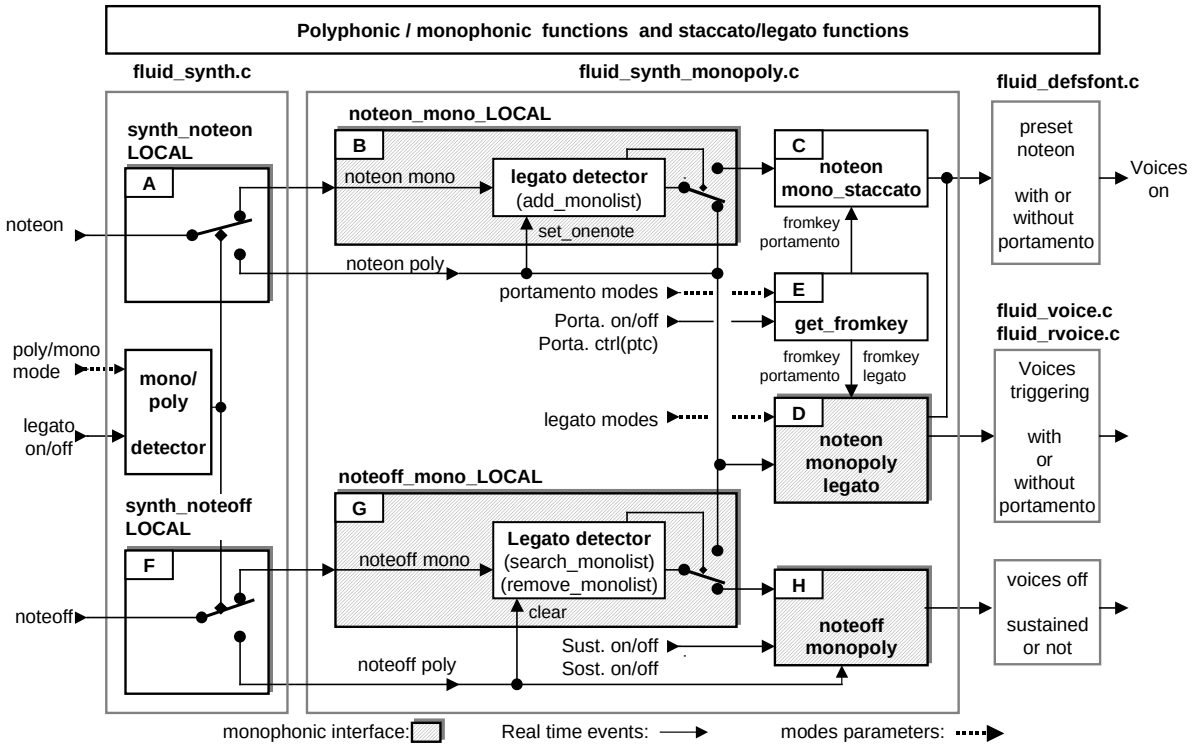


Fig.1: polyphonic and monophonic functions in FluidSynth.

The figure Fig.1 shows both the polyphonic and monophonic functions.
Please note the functions interface (greyed) of the whole monophonic logic.

MIDI noteOn/noteOff messages are received by **fluid_synth_noteon_LOCAL()** function (labelled **A**) and **fluid_synth_noteoff_LOCAL()** function (labelled **F**).

Depending on the poly mode and legato switch state (see 3.1.1, 3.1.2):

- A **noteon** event will be passed to **fluid_synth_noteon_mono_LOCAL()** (labelled **B**) if the channel is in monophic state (4.2, 4.3) or to **fluid_synth_noteon_monopoly_legato()** (labelled **D**) if the channel is in polyphonic state (see note).
- Similarly, a **noteoff** event will be passed to **fluid_synth_noteoff_mono_LOCAL ()** (labelled **G**) if the channel is in monophic state (4.4) or to **fluid_synth_noteoff_monopoly()** (labelled **H**) if the channel is in polyphonic state (4.5).

Note: There are situations where a noteon poly needs to be played legato (this is dependant of CC portamento control (PTC). More on this later in chapter 4.3).

4.2. Monophonic noteon and staccato functions

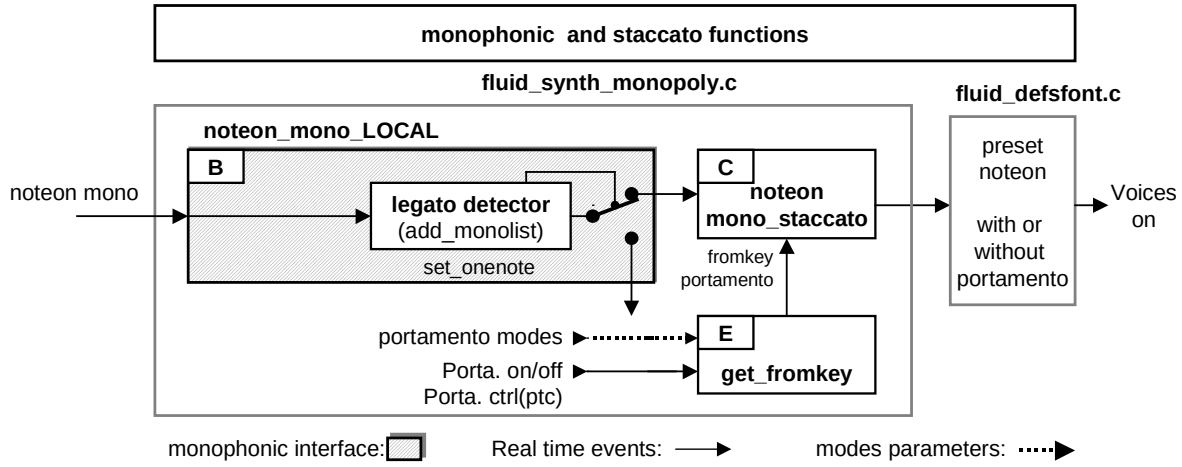


Fig.2: Monophonic and noteon staccato functions in FluidSynth.

When a **noteon** mono is received by **fluid_synth_noteon_mono_LOCAL()** function (labelled **B**) it uses the legato detector (see 3.4.3) to determine if the note must be played staccato or legato.

Figure 2 show the staccato situation (where no previous note have been depressed). **noteon** mono event is passed to **fluid_synth_noteon_mono_staccato()** function (labelled **C**).

Before the note been passed to **fluid_preset_noteon(), fluid_synth_noteon_mono_staccato()** must determine the **fromkey_portamento** parameter used by fluid_preset_noteon().

fromkey_portamento is returned by **fluid_synth_get_fromkey_portamento_legato()** function (labelled **E**). fromkey_portamento is set to valid/invalid key value depending of the **portamento modes** (see 3.1.7, 3.8), **CC portamento On/Off** (see 3.4.9), and **CC portamento control** (PTC) (see 3.4.11).

If **fromkey_portamento** is a valid key, a portamento effect is started inside **fluid_preset_noteon()** (see fluid_voice_calculate_runtime_synthesis_parameters()).

4.3. Monophonic-polyphonic noteon and legato functions

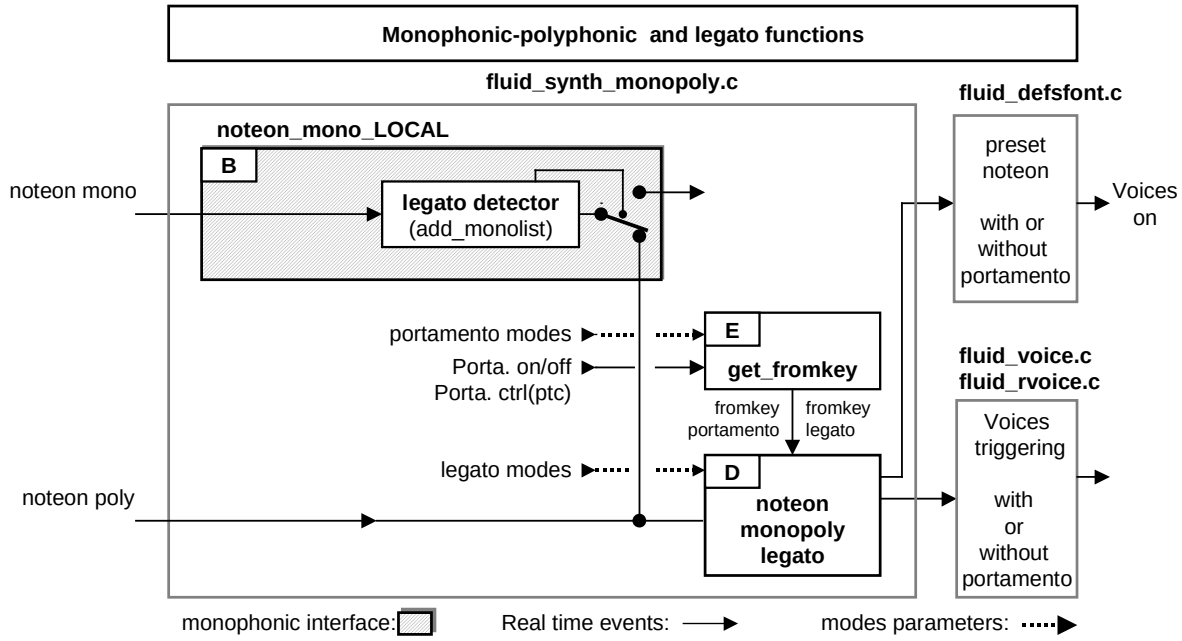


Fig.3: Monophonic and noteon legato functions in FluidSynth.

(a)When a **noteon** mono is received by **fluid_synth_noteon_mono_LOCAL()** function (labelled **B**) it uses the legato detector (see 3.4.3) to determine if the note must be played staccato or legato.

Figure 3 show the legato situation (where a previous note have been depressed). the noteon mono is passed to **fluid_synth_noteon_monopoly_legato()** function (labelled **D**).

fluid_synth_noteon_mono_legato() must determine the **fromkey_portamento** and **fromkey_legato** parameters used by fluid_preset_noteon() function or the voices triggering functions.

fromkey_portamento and **fromkey_legato** are returned by **fluid_synth_get_fromkey_portamento_legato()** function (labelled **E**). fromkey_portamento is set to valid/invalid key value depending of the **portamento modes** (see 3.1.7, 3.8) , **CC portamento On/Off** (see 3.4.9), and **CC portamento control** (PTC) (see 3.4.11).

Then depending of the legato modes (see 3.1.4) **fluid_synth_noteon_mono_legato()** will call the appropriate triggering functions:

- **fluid_voice_release()** , (legato mode 0)
- **fluid_voice_update_multi_retrigger_attack()**, (legato mode 1)

(b)Similarly a **noteon** poly is passed to **fluid_synth_noteon_monopoly_legato()** function (labelled **D**) and the explanations above are relevant. This allows an opportunity to get this note played legato with a previous note if a CC PTC have been received before this noteon. This behaviour is a MIDI specification (see 3.4.11 for details).

4.4. Monophonic noteoff and legato functions

When a **noteoff** mono is received by **fluid_synth_noteoff_mono_LOCAL()** function (labelled **G**) it uses the legato detector (see 3.4.3) to determine if the note must be played staccato or legato.

The situation is the similar as on noteOn mono (see 4.3 - a) except that now **fluid_synth_noteon_monopoly_legato()** function (labelled **D**) is called by fluid_synth_noteoff_mono_LOCAL().

4.5. Monophonic-polyphonic noteoff functions

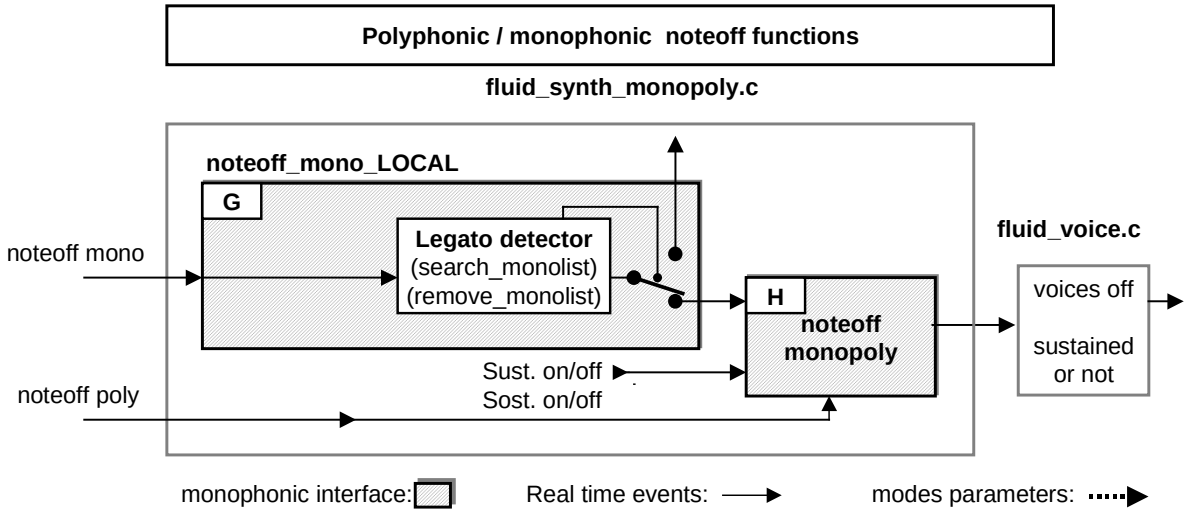


Fig.4: Monophonic – polyphonic noteoff functions in FluidSynth

When a **noteoff** mono is received by **fluid_synth_noteoff_mono_LOCAL()** function (labelled **G**) it uses the legato detector (see 3.4.3) to determine if the note must be played staccato or legato.

Figure 4 show the staccato situation (where a no previous note have been depressed). the noteoff mono is passed to **fluid_synth_noteoff_monopoly()** function (labelled **H**).

fluid_synth_noteoff_monopoly() have the same behaviour when the **noteoff** is **poly** of **mono**, except that for mono noteoff, if any pedal (sustain or sostenuto) is depressed, the key is memorized. This is neccessary when the next mono note will be played staccato, as any current mono note currently sustained should be released (see **fluid_synth_noteon_mono_staccato()**).

Note also that for a monophonic legato passage, the function is called only when the last noteoff of the passage occurs. That means that if sustain or sostenuto is depressed, only the last note of a legato passage will be sustained.

4.6. The legato detector

The legato detector is composed as this:

variables:

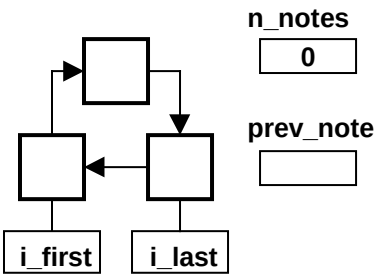
- **monolist** : circular buffer to store noteOn event in a circular manner
- **prev_note**: to store the most recent note before adding on noteon or before removing on noteoff
- **FLUID_CHANNEL_LEGATO_PLAYING** :legato/staccato state bit that informs on legato or staccato playing.

and functions

- **fluid_channel_add_monolist()**, for inserting a new note
- **fluid_channel_search_monolist()**, for searching the position of a note into the list.
- **fluid_channel_remove_monolist()**, for removing a note from the list.

Each MIDI channel have a legato detector.

4.6.1. The monophonic list: circular buffer **monolist**



Tfe monophonic list at initialization.

The list allows an easy automatic detection of a legato passage when it is played on a MIDI keyboard input device.
It is useful also when the input device is an ewi (electronic wind instrument) or evi (electronic valve instrument) and these instruments ar unable to send MIDI CC legato on/off.

The list remembers the notes in playing order.

- On noteOn n2, if a previous note n1 exists, there is a legato detection with n1 (with or without portamento from n1 to n2 .See note below).

- On noteOff of the running note n2, if a previous note n1 exists, there is a legato detection from n2 to n1, allowing fast trills playing (with or without portamento from n2 to n1. See note below).

Notes in the list are inserted to the end of the list that works like a circular buffer.The features are:

- 1) It is always possible to play an infinite legato passage in direct order (n1_On,n2_On,n3_On,....).

- 2) Playing legato in the reverse order (n10_Off, n9_Off,,...) helps in fast trills playing as the list memorizes 10 most recent notes.

- 3) Playing an infinite lagato passage in ascendant or descendant order, without playing trills is always possible using the usual way like this:
  First we begin with an ascendant passage:
  n1On, (n2On,n1Off), (n3On,n2Off), (n4On,n3Off),

  then we continue with a descendant passage
  (n3On,n4off), (n2On,n3off), (n1On,n2off), n1Off...and so on

Note about the size of monolist: **FLUID_CHANNEL_SIZE_MONOLIST**
The number of elements in monolist is defined **by FLUID_CHANNEL_SIZE_MONOLIST**.

- 1 is the minimum. it allows playing legato passage of any number of notes on noteOn only.
- Size above 1 allows playing legato on noteon but also on noteOff .This allows the musician to play fast trills. This feature is particularly useful when the MIDI input device is a keyboard.

A size of 10 is sufficient (because most musicians have only 10 fingers when playing a monophonic instrument).

Note:
Portamento is a feature independant of the legato detector. So portamento isn't part of the legato detector. However portamento (when enabled) is triggered at noteOn (like legato). Like in legato situation it is usual to have a portamento from a note ***'fromkey'*** to another note ***'tokey'***. Portamento fromkey note choice is determined at noteOn by **fluid_synth_get_fromkey_portamento_legato()** (see 4.2, 4.3).

4.6.2. The previous note: **prev_note**

prev_note is used to determine **fromkey_portamento** as well as **fromkey_legato** (see **fluid_synth_get_fromkey_portamento_legato()**, see 4.2, 4.3).

prev_note is updated on noteOn/noteOff mono by the legato detector as this:

- On **noteOn mono**, before adding a new note into the monolist, the most recent note in the list (i.e at **i_last** position) is kept in **prev_note**.
- Similarly, on **noteOff mono** , before removing a note out of the monolist, the most recent note (i.e at **i_last** position) is kept in **prev_note**.

4.6.3. legato/staccato state: **FLUID_CHANNEL_LEGATO_PLAYING**

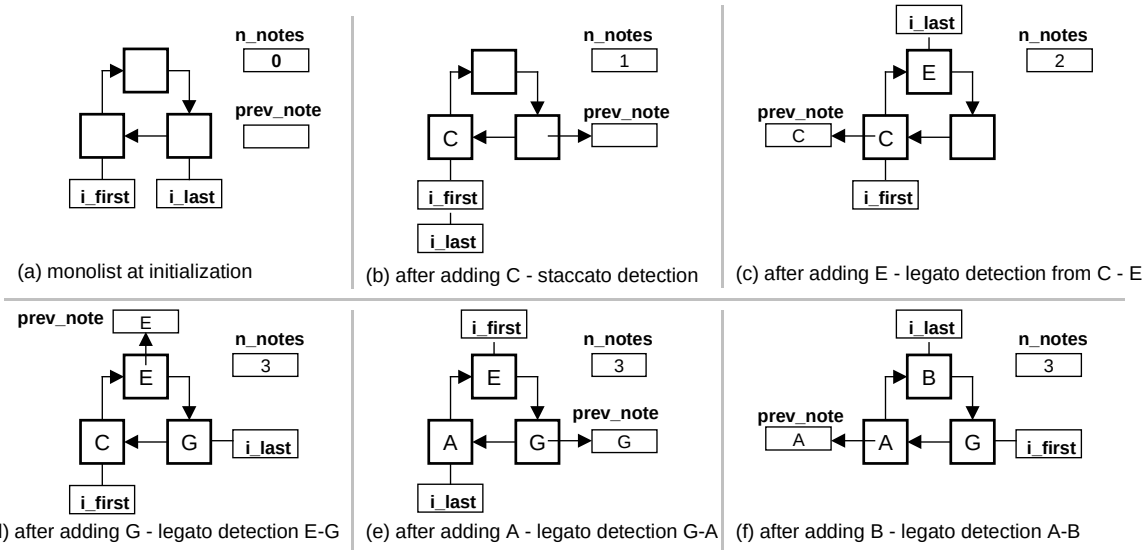
FLUID_CHANNEL_LEGATO_PLAYING bit of channel mode keeps trace of the legato /staccato state playing:

- On noteOn, this state is used by fluid_synth_noteon_mono_LOCAL() to play the current note legato or staccato.
- On noteOff, this state is used by fluid_synth_noteoff_mono_LOCAL() to play the current noteOff legato with the most recent note.

FLUID_CHANNEL_LEGATO_PLAYING bit is updated on noteOn/noteOff mono by the legato detector:

- On **noteOn**, before inserting a new note into the monolist.
- On **noteOff**, after removing a note out of the monolist.

4.6.4. Adding notes in monolist: **fluid_channel_add_monolist()**



The monophonic list is a circular buffer of SIZE_MONOLIST elements. The figure shows a list with 3 elements only. Each element is linked forward at initialisation time.

- When a note is added at noteOn each element is used in the forward direction and indexed by **i_last** variable.
- **prev_note** keeps a trace of the note prior **i_last note**.
- The most recent note added is indexed by **i_last.**
- The most ancient note added is the first note indexed by **i_first**. i_first is moving in the forward direction in a circular manner.

(b) On noteon C:

- there are no previous note (prev_note is INVALID_NOTE).
- there is a staccato detection.

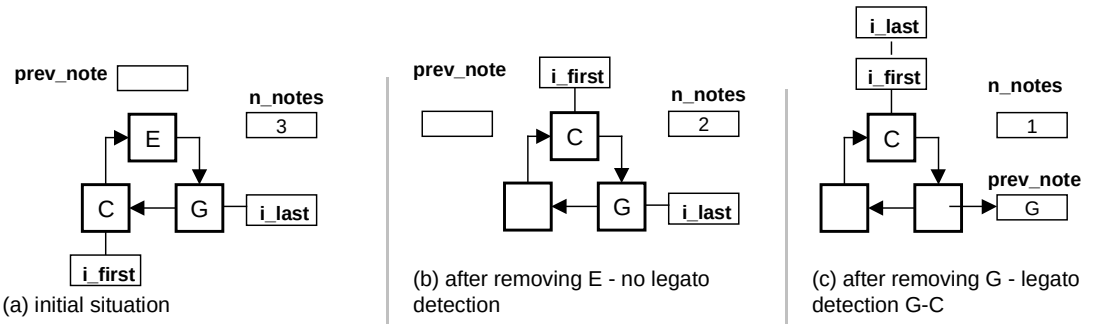
(c) On noteon E:
- previous note = C.
- there is a legato detection from C to E.

(d) On noteon G:
- previous note = E.
- there is a legato detection from E to G.

(e) On noteon A:
- previous note = G.
- there is a legato detection from G to A.

(f) On noteon B:
- previous note = A.
- there is a legato detection from A to B.

4.6.5. Removing notes in monolist: **fluid_channel_remove_monolist()**



when a note is removed at noteOff the element concerned is unlinked and relinked after the **i_last** element.
- If the note to be removed is **i_last** , **prev_note** keeps a trace of the note removed.

(b) On noteoff E:
- E is not the last note so there is no legato detection.

(c) On noteoff G:
- G is the last note so, prev_note = G .
- there is a legato detection from G to C.

(d) On noteoff A:
- G is the last note so prev_note = A .
- there is a no legato detection.

4.7. Legato playing through Instrument Zone and Preset Zone in Soundfont

This patch now accepts legato passage through multiples Instruments Zones and Preset Zone.This chapter describes the enhancement to suppress the know (and unacceptable) limitations of previous patches.

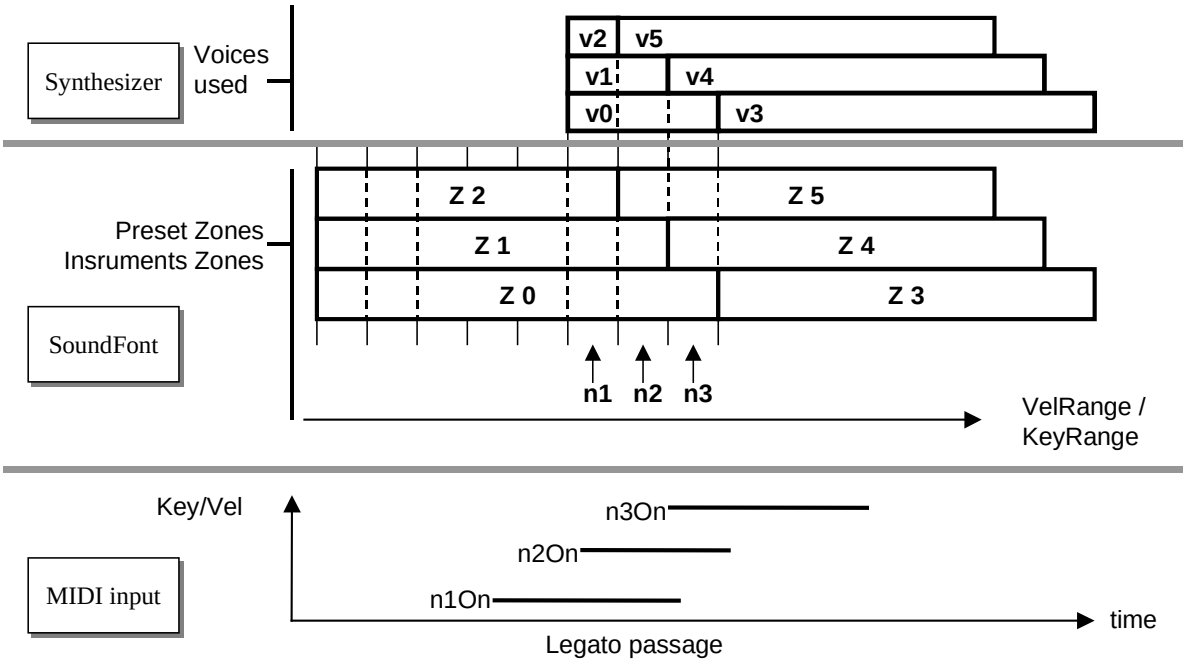


Fig.1: legato passage and zones switching.

Soundfont preset are often designed with multiple key ranges and velocity ranges at Instrument Level. That means that when playing a monophonic passage of notes n1,n2,n3 in a legato manner. n1 been the first:

1) following notes n2, n3 each make use of n1 voices, if n2 or n3 are still in the KeyRange, VelRange of Instrument Zone or Preset Zone (IZ,PZ) of the n1 running voices.
2) If n2,n3 are outside VelRange and KeyRange of some running voices, those voices are forced in release section.
3) Next, new eligible voices for n2,n3 need to be allocated.

So, on noteOn, the legato logic must do two things:

- the legato job (step 1 above) and also
- it must respect the keyRange or velRange switching as instructed by the SoundFont designer (steps 2 and 3 above).
  If a zone switching occurs it can compromise the legato job but this is necessary to respect the Soundfont designer decision. Previous patch didn't know the zone switching, making use of previous voices even the voice pitch gets out the keyRange of the zone. This produced a sound very unatural and unacceptable.

Exemple:
Let n1,vel1 (note and velocity) with layered zones Z0,Z1,Z2 each represented by their respective VelRange,KeyRange (instructed by the soundfont Preset Zone/Instrument Zone).See Fig.1

- At n1On, voices v0,v1,v2 are allocated and started for note n1.
- At n2On , assuming that a legato passage as been detected, the following occurs:
  1) n2,vel2 is found in the VelRange,KeyRange of Z0, so voice v0 is used (without allocation) to play n2.

2) n2 is found in the VelRange,KeyRange of zone Z1.This is the same situation as (1) so voice v1 is used (without allocation) to play n2 .
3) n2 is found outside the VelRange,KeyRange of Z2.Voice v2 can't be used to play n2, so voice v2 is forced in release section (see note 2 below).
4) Now all the previous voices of n1 have been explored. However, it is possible that n2,vel2 will be able to enter in the Velrange,KeyRange of a new Preset Zones or Instrument Zones (Z5).(see note 1) In this case new voices (v5) need to be allocated to play this new Preset Zones or Instrument Zones (Z5).

This allocation is done using fluid_defpreset_noteon(). During this new allocation, the Instrument Zones corresponding to the already running voices v0,v1 (revived at step 1 and 2) must be ignored. So, steps 1 and 2 need to mark the corresponding Instrument Zone (Z0,Z1) to be ignored during step 3.

Note 1: This situation occurs when Instruments Zones of each layer overlaps partially (i.e the beginning/ending of one zone doesn't coincide with the beginning/ending of others zones) as show in figure 1.

Note 2:
As some voices are necessarily forced in release section, the passage loses its legato caracter at the Instrument Zone transition time, that is:

- The legato mode 1 is retrograded to legato mode 0.
- This can be more compensed also at soundfont design level when a Preset Zone have more than one Instruments Zones that overlapps partially as this kind of soundfont design enhance the blending of voices and legato rendering (see Note 1).