pi-loop

0.9(prototype)

Generated by Doxygen 1.8.17

1 Main Page 1

2 Class Index 5

2.1 Class List . . . 5

3 Class Documentation 7

3.1 AudioServer Class Reference . . . 7

3.1.1 Detailed Description . . . 7

3.1.2 Member Function Documentation . . . 7

3.1.2.1 start() . . . 7

3.1.2.2 stop() . . . 7

3.2 Buttons Class Reference . . . 8

3.2.1 Detailed Description . . . 8

3.3 Channel Class Reference . . . 8

3.3.1 Detailed Description . . . 9

3.3.2 Member Function Documentation . . . 9

3.3.2.1 add_track() . . . 9

3.3.2.2 clean() . . . 9

3.3.2.3 dub() . . . 9

3.3.2.4 get_name() . . . 9

3.3.2.5 get_num_tracks() . . . 10

3.3.2.6 get_out_signal() . . . 10

3.3.2.7 get_volume() . . . 10

3.3.2.8 isEmpty() . . . 10

3.3.2.9 rec() . . . 10

3.3.2.10 reset() . . . 11

3.3.2.11 reset_num_tracks() . . . 11

3.3.2.12 set_name() . . . 11

3.3.2.13 set_volume() . . . 11

3.3.2.14 undub() . . . 11

3.3.2.15 update_rec_signal() . . . 11

3.3.2.16 volume_down() . . . 12

3.3.2.17 volume_up() . . . 12

3.4 Config Class Reference . . . 12

3.4.1 Detailed Description . . . 13

3.4.2 Member Function Documentation . . . 13

3.4.2.1 display() . . . 13

3.4.2.2 get_button_state() . . . 13

3.4.2.3 get_curr_session() . . . 13

3.4.2.4 get_max_sessions() . . . 13

3.4.2.5 open() . . . 14

3.4.2.6 reset() . . . 14

3.4.2.7 save() . . . 14

3.5 Effects Class Reference . . . 14
3.5.1 Detailed Description . . . 15
3.5.2 Member Function Documentation . . . 15
3.5.2.1 apply() . . . 15
3.5.2.2 initialize_effects() . . . 15
3.5.2.3 toggle_effect() . . . 15
3.6 Handshake Class Reference . . . 16
3.6.1 Detailed Description . . . 16
3.6.2 Member Function Documentation . . . 17
3.6.2.1 activate() . . . 17
3.6.2.2 check_status() . . . 17
3.6.2.3 connect_input_device() . . . 17
3.6.2.4 connect_output_device() . . . 17
3.6.2.5 disconnect_client() . . . 17
3.6.2.6 disconnect_input_device() . . . 18
3.6.2.7 disconnect_output_device() . . . 18
3.6.2.8 get_input_buffer() [1/2] . . . 18
3.6.2.9 get_input_buffer() [2/2] . . . 18
3.6.2.10 get_output_buffer() [1/2] . . . 19
3.6.2.11 get_output_buffer() [2/2] . . . 19
3.6.2.12 link_client() . . . 19
3.6.2.13 prevent_failure() . . . 20
3.6.2.14 register_devices() . . . 20
3.6.2.15 register_input_port() . . . 20
3.6.2.16 register_output_port() . . . 20
3.6.2.17 set_buffer_size() . . . 20
3.6.2.18 set_process_callback() . . . 21
3.7 hardwareInterface Class Reference . . . 21
3.7.1 Detailed Description . . . 21
3.7.2 Member Function Documentation . . . 21
3.7.2.1 deafen() . . . 21
3.7.2.2 display() . . . 22
3.7.2.3 get_display_initializer() . . . 22
3.7.2.4 get_metro_display() . . . 22
3.7.2.5 listen() . . . 22
3.7.2.6 update() . . . 23
3.8 Keyboard Class Reference . . . 23
3.8.1 Detailed Description . . . 23
3.8.2 Member Function Documentation . . . 23
3.8.2.1 is_pressed() . . . 23
3.9 Leds Class Reference . . . 24
3.9.1 Detailed Description . . . 24

3.10 Looper Class Reference . . . 24
3.10.1 Detailed Description . . . 25
3.10.2 Member Function Documentation . . . 25
3.10.2.1 display_states() . . . 25
3.10.2.2 get_metronome_state() . . . 25
3.10.2.3 recdub() . . . 25
3.10.2.4 reset() . . . 26
3.10.2.5 save() . . . 26
3.10.2.6 start_stop_all() . . . 26
3.10.2.7 stoperase() . . . 26
3.10.2.8 tap_alter_metronome() . . . 27
3.10.2.9 update_buffer() . . . 27
3.10.2.10 volume_change() . . . 27
3.10.2.11 volume_down() . . . 28
3.10.2.12 volume_up() . . . 28
3.11 LooperState Struct Reference . . . 28
3.11.1 Detailed Description . . . 29
3.12 Menu Class Reference . . . 29
3.12.1 Detailed Description . . . 29
3.12.2 Member Function Documentation . . . 29
3.12.2.1 load_session() . . . 29
3.12.2.2 notify_menu() . . . 30
3.12.2.3 set_display_initializer() . . . 30
3.12.2.4 set_metro_display() . . . 30
3.12.2.5 setup() . . . 30
3.12.2.6 unload() . . . 31
3.13 Metronome Class Reference . . . 31
3.13.1 Detailed Description . . . 31
3.13.2 Member Function Documentation . . . 31
3.13.2.1 alter_tempo() . . . 32
3.13.2.2 clear() . . . 32
3.13.2.3 lock() . . . 32
3.13.2.4 pause() . . . 32
3.13.2.5 start_timing() . . . 32
3.13.2.6 stop_timing() . . . 32
3.13.2.7 sync() . . . 33
3.13.2.8 tap_tempo() . . . 33
3.13.2.9 tick_tock() . . . 33
3.13.2.10 unlock() . . . 33
3.13.2.11 unpause() . . . 33
3.14 Mixer Class Reference . . . 34
3.14.1 Detailed Description . . . 34

3.14.2 Member Function Documentation . . . 34
3.14.2.1 save_jam() . . . 34
3.14.2.2 update_buffer() . . . 34
3.15 Monitor Class Reference . . . 35
3.15.1 Detailed Description . . . 35
3.15.2 Member Function Documentation . . . 35
3.15.2.1 connect() . . . 35
3.15.2.2 disconnect() . . . 36
3.15.2.3 get_states() . . . 36
3.15.2.4 initialize_effects() . . . 36
3.15.2.5 mute_input() . . . 36
3.15.2.6 mute_output() . . . 37
3.15.2.7 process() . . . 37
3.15.2.8 set_stream_buffer() . . . 37
3.15.2.9 toggle_effect() . . . 37
3.15.2.10 toggle_states() . . . 38
3.15.2.11 update_states() . . . 38
3.16 PiLoop Class Reference . . . 38
3.16.1 Detailed Description . . . 39
3.16.2 Member Function Documentation . . . 39
3.16.2.1 start_console() . . . 39
3.17 Potentiometers Class Reference . . . 39
3.17.1 Detailed Description . . . 39
3.18 Response Struct Reference . . . 39
3.18.1 Detailed Description . . . 40
3.18.2 Member Function Documentation . . . 40
3.18.2.1 isEmpty() . . . 40
3.18.2.2 reset() . . . 40
3.19 Screen Class Reference . . . 40
3.19.1 Detailed Description . . . 41
3.19.2 Member Function Documentation . . . 41
3.19.2.1 display() . . . 41
3.19.2.2 initialize_states() . . . 41
3.19.2.3 perform_operation() . . . 41
3.19.2.4 tick_tock() . . . 42
3.19.2.5 turnOff() . . . 42
3.20 Session Class Reference . . . 42
3.20.1 Detailed Description . . . 43
3.20.2 Member Function Documentation . . . 43
3.20.2.1 evacuate() . . . 43
3.20.2.2 get_name() . . . 43
3.20.2.3 load() . . . 43

3.20.2.4 migrate() . . . 43
3.20.2.5 notify_session() . . . 44
3.20.2.6 save() . . . 44
3.20.2.7 save_jam() . . . 44
3.20.2.8 set_disp_initializer() . . . 44
3.20.2.9 set_metronome_display() . . . 44
3.20.2.10 set_name() . . . 45
3.20.2.11 setup() . . . 45
3.21 time_signature Struct Reference . . . 45
3.21.1 Detailed Description . . . 45
3.22 Trigger Struct Reference . . . 46
3.22.1 Detailed Description . . . 46
3.22.2 Member Function Documentation . . . 46
3.22.2.1 reset() . . . 46
3.23 UI Class Reference . . . 46
3.23.1 Detailed Description . . . 47
3.23.2 Member Function Documentation . . . 47
3.23.2.1 initialize_display() . . . 47
3.23.2.2 listen_user() . . . 47
3.23.2.3 metro_display() . . . 47
3.23.2.4 show() . . . 48
3.23.2.5 turnOff() . . . 48
3.23.2.6 update_output_state() . . . 48

Index **49**

Chapter 1

Main Page

A linux based audio looper written in C++.

Description | Features | Documentation | Installation | Usage | Future_Work | Feedback | Related

Description

Pi-loop is a real-time audio looper application that enables users to create interactive musical sessions. One may use an external audio interface to connect a microphone and an instrument or to use the integrated sound card to test it.

This repo consists of 2 different **modes** that make use of alternative user interfaces to interact with the software:

- **PC mode** (version 0.9)
  This version runs directly on a Linux machine and allows the usage of **computer components** such as the **keyboard** ∗**to trigger events**∗∗∗ **and the** ∗∗**screen** ∗**to access the output state of the program**∗∗∗**.**
- ∗∗**RPI mode** (will be included in the next update)
  This version runs on a **Raspberry Pi**∗∗ and uses **GPIO**-based connected buttons and rotaries to enable user interaction with the program, and LEDs to access the output state of the program. This mode is currently under development.∗

Both modes lie on the same codebase and can be configured on build. More information can be found in the installation section below.

Features

Key features that make Piloop unique:

- **Input signal handling flexibility**

  Individual ARM and Monitor (IN/OUT) states for each input channel allow users to choose on the fly which input channels will be streamed on the speakers and which will contribute to the creation of the current loop.

- **Jam saving feature**

  Once decided, users can save their overall musical session as a wave file, by pressing a single button. Therefore, not only the loop stream is saved but the combined audio signal stream that has been streamed on the speakers during the session. This can be quite handy for musicians who want to save a session and use it to keep track of their inspirational dynamics or alternatively to use the saved file as a songwriting guide that can be further manipulated in a DAW afterward.

- **Individual effects on each input channel**

  This increases the flexibility to add effects individually on each input channel.

Common features supported:

- 2 input channels∗
- 1 stereo output channel∗
- 3 fully operational loop channels (read further in the manual page)
- metronome
  - tap tempo
  - alter tempo
- a menu feature to load, change, and save session presets.

∗∗depends on the number of IO channels supported by the audio interface used.∗

Documentation

Documentation file is uploaded in a `pdf` form.

To use an html version, install and run doxygen to generate the documentation files.

```
cd pi-loop-console
doxygen Doxyfile
```

The documentation files will be stored within `files/docs/html` directory.

If you like, you may also check the `UML file`.

Installation

Follow this link to access information on *configuring*, *building*, *running*, and *using* PiLoop in **PC mode**.

For the **RPI mode**, behold for the upcoming updates. You may follow this link to obtain an overview on the connections made to the GPIO pins.

Usage

To run Piloop, you have to run with sudo priviledge:
```
$ sudo ./piloop
```

To see how to use it visit the manual page

Future_Work

The future work listing is summarized as follows:

- Fix found bugs.
- Improve existing features
- Add new features
- conclude Raspeberry Pi setup
- The name of this application is Piloop and not without reason. The upcoming Piloop-1.0 is to run on a headless Raspberry Pi. Details about the current setup can be found here.

Visit the future-work list to get informed about what's next.

Feedback

If you have used this application and would like to contribute or share your feedback and recommendations on improvements, or if there is anything you would like to say about it by any means, please don't hesitate to share it on `Gitter` or contact me at `melissaspaschalis@gmail.com`.

Related

If you liked this project, you may also like:

- `Raspberry-Pi-Looper-synth-drum-thing`
- `making-a-looper-with-pure-data/`
- `iav` : An interactive audio visualizer

:smiley: :notes: :musical_note: :headphones: :saxophone: :accordion: :musical_keyboard: :violin: :banjo: ↩ :trumpet: :guitar: :desktop_computer: :drum:

Enjoy Piloop

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AudioServer
: The jack-audio server running on the alsa drivers . . . 7

Buttons
: Buttons as the GPIO-based input interface . . . 8

Channel
: Defines a looper-channel designed to loop tracks . . . 8

Config
: The config module responsible for saving, loading, and initializing session data . . . 12

Effects
: The effects module. Currently unsupported . . . 14

Handshake
: The jack-audio client . . . 16

hardwareInterface
: The generic interface of the pi-loop. It has an input and an output interface . . . 21

Keyboard
: The keyboard as the computer-based input interface . . . 23

Leds
: LEDs as the GPIO-based output interface . . . 24

Looper
: Constisted of 3 loop-channels . . . 24

LooperState
: Stores the looper state . . . 28

Menu
: The programs menu that acts as the session manager . . . 29

Metronome
: A metronome defined within Looper . . . 31

Mixer
: The mixer module responsible to stream the output signal . . . 34

Monitor
: The monitor module encapsulates the jack audio client . . . 35

PiLoop
: The threading manager. It starts the program . . . 38

Potentiometers
: Potentiometers to adjust the looper-channel's volume . . . 39

Response
: Defines a response message destinated to the displayer module of the program . . . 39

Screen
: The screen as the computer-based output interface . . . 40

Session
: Defines a musical session. It is consisted of a Monitor, a Looper and a Mixer . . . 42

time_signature
Struct to hold the information of rythm 45
Trigger
Holds the triggers of the user 46
UI
Defines a computer based interface constisted of a keyboard and the screen 46

Chapter 3

Class Documentation

3.1 AudioServer Class Reference

The jack-audio server running on the alsa drivers.
`#include <audioserver.h>`

Public Member Functions

- AudioServer (const char *driverName=supported_driver)

  Default constructor.
- void start ()

  Starts the jack audio server.
- void stop ()

  Stops the jack audio server.

3.1.1 Detailed Description

The jack-audio server running on the alsa drivers.

3.1.2 Member Function Documentation

3.1.2.1 start()

`void AudioServer::start ( )`
Starts the jack audio server.

Returns

void

3.1.2.2 stop()

`void AudioServer::stop ( )`
Stops the jack audio server.

Returns

void

The documentation for this class was generated from the following files:

- include/audioserver.h
- src/audioserver.cpp

3.2 Buttons Class Reference

Buttons as the GPIO-based input interface.
`#include <buttons.h>`

Public Member Functions

- void **setup** (int)
- void **is_pressed** (Trigger &)

3.2.1 Detailed Description

Buttons as the GPIO-based input interface.
The documentation for this class was generated from the following files:

- include/gpio/buttons.h
- src/gpio/buttons.cpp

3.3 Channel Class Reference

Defines a looper-channel designed to loop tracks.
`#include <channel.h>`

Public Member Functions

- Channel ()

  default contructor
- void reset ()

  function used to clear all data written on the channels.
- void set_name (const char *)

  function used to set a name on the current channel.
- const char * get_name ()

  function used to get the name of the current channel.
- void add_track ()

  function used to add a new track on the current channel. In fact, it only stores the information that the track is saved.
- void reset_num_tracks ()

  sets num tracks to zero.
- int get_num_tracks ()

  gets the number of tracks of this channel.
- void update_rec_signal (float[BUFFER_SIZE], int)

  function to pass the current buffer to rec or to dub. This function is only called when there is work to be done.
- float get_volume ()

  gets the volume of the current channel
- void set_volume (int)

  sets the volume for the current channel
- void volume_up ()

  Used in PC mode to increase the volume of channel by <volumeStep> percent.
- void volume_down ()

  Used in PC mode to decrease the volume of channel by <volumeStep> percent.
- void clean ()

  function used to clean recorded data
- void rec (float[BUFFER_SIZE], int)

  function to record data. It is used when channel has no tracks on it yet.
- void dub (float[BUFFER_SIZE], int)

function to store new data on top of previously recorded data.

- void undub ()

  function used to erase last channel's record.

- std::vector< float > get_out_signal (int)

  function to return the channels output signal

- bool isEmpty ()

  function to return whether the channel has recorded tracks or not

3.3.1 Detailed Description

Defines a looper-channel designed to loop tracks.

3.3.2 Member Function Documentation

3.3.2.1 add_track()

```
void Channel::add_track ( )
```

function used to add a new track on the current channel. In fact, it only stores the information that the track is saved.

Returns

void

3.3.2.2 clean()

```
void Channel::clean ( )
```

function used to clean recorded data

Returns

void

3.3.2.3 dub()

```
void Channel::dub (
            float monitor[BUFFER_SIZE],
            int idx )
```

function to store new data on top of previously recorded data.

Returns

void

3.3.2.4 get_name()

```
const char * Channel::get_name ( )
```

function used to get the name of the current channel.

Returns

void

3.3.2.5 get_num_tracks()

```
int Channel::get_num_tracks ( )
```
gets the number of tracks of this channel.

Returns

int - the number of tracks

3.3.2.6 get_out_signal()

```
std::vector< float > Channel::get_out_signal (
            int idx )
```
function to return the channels output signal

Parameters

int	- the index that indicates the current playback position

Returns

std::vector<float> - the output buffer

3.3.2.7 get_volume()

```
float Channel::get_volume ( )
```
gets the volume of the current channel

Returns

float - the volume as type of a float percentage

3.3.2.8 isEmpty()

```
bool Channel::isEmpty ( )
```
function to return whether the channel has recorded tracks or not

Returns

bool - true if has tracks, otherwise false.

3.3.2.9 rec()

```
void Channel::rec (
            float monitor[BUFFER_SIZE],
            int idx )
```
function to record data. It is used when channel has no tracks on it yet.

Returns

void

3.3.2.10 reset()

```
void Channel::reset ( )
```

function used to clear all data written on the channels.

Returns

void

3.3.2.11 reset_num_tracks()

```
void Channel::reset_num_tracks ( )
```

sets num tracks to zero.

Returns

void

3.3.2.12 set_name()

```
void Channel::set_name (
            const char * name )
```

function used to set a name on the current channel.

Returns

void

3.3.2.13 set_volume()

```
void Channel::set_volume (
            int volume )
```

sets the volume for the current channel

Parameters

int	- the volume as type of an int percentage

Returns

void

3.3.2.14 undub()

```
void Channel::undub ( )
```

function used to erase last channel's record.

Returns

void

3.3.2.15 update_rec_signal()

```
void Channel::update_rec_signal (
            float monitor[BUFFER_SIZE],
            int idx )
```

function to pass the current buffer to rec or to dub. This function is only called when there is work to be done.

Returns

void

3.3.2.16 volume_down()

`void Channel::volume_down ( )`
Used in PC mode to decrease the volume of channel by <volumeStep> percent.

Returns

void

3.3.2.17 volume_up()

`void Channel::volume_up ( )`
Used in PC mode to increase the volume of channel by <volumeStep> percent.

Returns

void

The documentation for this class was generated from the following files:

- include/channel.h
- src/channel.cpp

3.4 Config Class Reference

The config module responsible for saving, loading, and initializing session data.
`#include <config.h>`

Public Member Functions

- void open (int)
  Reads the information stored within the configuration file for a single session.
- void save ()
  Saves the information of the current session within the configuration file.
- void display ()
  Displays the information of the current session.
- void reset ()
  Discards all the changes made during a session.
- bool get_button_state (const Control &)
  getter for button states. Used only for initialization.
- void toggle_button_state (const Control &)
  Toggles the button states. Discarded since not thread safe.
- int get_max_sessions ()
  Method to read the number of session defined within the configuration file.
- int get_curr_session ()
  Method to return the id of the current session.

Static Public Member Functions

- static Config & getInstance ()
  Method to return a reference to the Singleton object.

Public Attributes

- std::string **currSession_name**
- std::unordered_map< Control, bool > **button_states**
- float **tempo**
- int **rythm_numerator**
- int **rythm_denominator**
- std::string **jam_savepath**
- std::string **session_savepath**

3.4.1 Detailed Description

The config module responsible for saving, loading, and initializing session data.

3.4.2 Member Function Documentation

3.4.2.1 display()

```
void Config::display ( )
```

Displays the information of the current session.

Returns

void

3.4.2.2 get_button_state()

```
bool Config::get_button_state (
            const Control & ctl )
```

getter for button states. Used only for initialization.

Parameters

const	Control& - the related button of interest

Returns

bool - the state of the current button

3.4.2.3 get_curr_session()

```
int Config::get_curr_session ( )
```

Method to return the id of the current session.

Returns

int - the id of the current session.

3.4.2.4 get_max_sessions()

```
int Config::get_max_sessions ( )
```

Method to read the number of session defined within the configuration file.

Returns

int - the maximum number of sessions.

3.4.2.5 open()

```
void Config::open (
            int curr_session )
```

Reads the information stored within the configuration file for a single session.

Parameters

int	- the session id.

Returns

void

3.4.2.6 reset()

```
void Config::reset ( )
```

Discards all the changes made during a session.

Returns

void

3.4.2.7 save()

```
void Config::save ( )
```

Saves the information of the current session within the configuration file.

Returns

void

The documentation for this class was generated from the following files:

- include/config.h
- src/config.cpp

3.5 Effects Class Reference

The effects module. Currently unsupported.

```
#include <effects.h>
```

Public Member Functions

- Effects ()

  Class contructor.
- void initialize_effects (const bool[F_NUM_INPUTS][NUM_EFFECTS])

  Implicit initialization of all effects for each channel, depending on the state of the current session.
- bool toggle_effect (int, int)

  Enables/Disables the an effect.
- void apply (float *[F_NUM_INPUTS])

  Function to apply all the enabled effects to the input signal altogether. Currently unsupported.

3.5.1 Detailed Description

The effects module. Currently unsupported.

3.5.2 Member Function Documentation

3.5.2.1 apply()

```
void Effects::apply (
            float * input_buffers[F_NUM_INPUTS] )
```

Function to apply all the enabled effects to the input signal altogether. Currently unsupported.

Parameters

float[F_NUM_INPUTS]*	- the input signal

Returns

void

3.5.2.2 initialize_effects()

```
void Effects::initialize_effects (
            const bool effects_curr_state[F_NUM_INPUTS][NUM_EFFECTS] )
```

Implicit initialization of all effects for each channel, depending on the state of the current session.

Parameters

const	bool[F_NUM_INPUTS][NUM_EFFECTS] - a boolean table containing the states of the effects (enabled/disabled) for each input channel.

Returns

void

3.5.2.3 toggle_effect()

```
bool Effects::toggle_effect (
            int ch,
            int eff )
```

Enables/Disables the an effect.

Parameters

int	- the corresponding channel
int	- the corresponding effect

Returns

bool - the current state of the corresponding effect after processing.

The documentation for this class was generated from the following files:

- include/effects.h
- src/effects.cpp

3.6 Handshake Class Reference

The jack-audio client.
`#include <handshake.h>`

Public Member Functions

- Handshake ()

  Class contructor.
- void link_client (char *)

  Function to open the jack client.
- void set_process_callback ()

  Function to set the callback function used for streaming the audio signal.
- void prevent_failure ()

  Function to set another function to be called in case the system fails.
- void register_input_port (int)

  Function to register a new input port.
- void register_output_port (int)

  Function to register a new output port.
- void activate ()

  Function to activate the jack client.
- void register_devices ()

  Function to register the input and the output devices.
- void set_buffer_size ()

  Function to change the buffer size using the preprocessor derivative used in audio_settings.h.
- void check_status ()

  Function to check the status of the client connection.
- void disconnect_client ()

  Function to disconnect the client from the server.
- void connect_input_device (int)

  Function to connect input ports with input devices.
- void connect_output_device (int)

  Function to connect output ports with output devices.
- void disconnect_input_device (int)

  Function to disconnect input ports from the connected input devices.
- void disconnect_output_device (int)

  Function to disconnect output ports from the connected output devices.
- float * get_input_buffer (int)

  Function to get the input buffer from a specific port-device.
- float * get_output_buffer (int)

  Function to get the output buffer to which the output port-device is connected.
- void get_input_buffer (int, float *)

  Function to get the input buffer from a specific port-device.
- void get_output_buffer (int, float *)

  Function to get the output buffer to which the output port-device is connected.

3.6.1 Detailed Description

The jack-audio client.

3.6.2 Member Function Documentation

3.6.2.1 activate()

```
void Handshake::activate ( )
```

Function to activate the jack client.

Returns

void

3.6.2.2 check_status()

```
void Handshake::check_status ( )
```

Function to check the status of the client connection.

Returns

void

3.6.2.3 connect_input_device()

```
void Handshake::connect_input_device (
            int idx )
```

Function to connect input ports with input devices.

Parameters

int	- index of the input port.

Returns

void

3.6.2.4 connect_output_device()

```
void Handshake::connect_output_device (
            int idx )
```

Function to connect output ports with output devices.

Parameters

int	- index of the output port.

Returns

void

3.6.2.5 disconnect_client()

```
void Handshake::disconnect_client ( )
```

Function to disconnect the client from the server.

Returns

void

3.6.2.6 disconnect_input_device()

```
void Handshake::disconnect_input_device (
            int idx )
```

Function to disconnect input ports from the connected input devices.

Parameters

int	- index of the input port.

Returns

void

3.6.2.7 disconnect_output_device()

```
void Handshake::disconnect_output_device (
            int idx )
```

Function to disconnect output ports from the connected output devices.

Parameters

int	- index of the output port.

Returns

void

3.6.2.8 get_input_buffer() [1/2]

```
float * Handshake::get_input_buffer (
            int idx )
```

Function to get the input buffer from a specific port-device.

Parameters

int	- index of the input port.

Returns

float* - the input buffer

3.6.2.9 get_input_buffer() [2/2]

```
void Handshake::get_input_buffer (
            int ,
            float *  )
```

Function to get the input buffer from a specific port-device.

Parameters

int	- index of the input port.
float∗	- the input buffer passed by value.

Returns

void

3.6.2.10 get_output_buffer() [1/2]

```
float * Handshake::get_output_buffer (
            int idx )
```

Function to get the output buffer to which the output port-device is connected.

Parameters

int	- index of the output port.

Returns

float∗ - the output buffer where the output signal is written to.

3.6.2.11 get_output_buffer() [2/2]

```
void Handshake::get_output_buffer (
            int ,
            float *  )
```

Function to get the output buffer to which the output port-device is connected.

Parameters

int	- index of the output port.
float∗	- the output buffer passed by value.

Returns

void

3.6.2.12 link_client()

```
void Handshake::link_client (
            char * name )
```

Function to open the jack client.

Parameters

char∗	- the client name

Returns

void

3.6.2.13 prevent_failure()

```
void Handshake::prevent_failure ( )
```

Function to set another function to be called in case the system fails.

Returns

void

3.6.2.14 register_devices()

```
void Handshake::register_devices ( )
```

Function to register the input and the output devices.

Returns

void

3.6.2.15 register_input_port()

```
void Handshake::register_input_port (
            int idx )
```

Function to register a new input port.

Parameters

int	- index of the input port.

Returns

void

3.6.2.16 register_output_port()

```
void Handshake::register_output_port (
            int idx )
```

Function to register a new output port.

Parameters

int	- index of the output port.

Returns

void

3.6.2.17 set_buffer_size()

```
void Handshake::set_buffer_size ( )
```

Function to change the buffer size using the preprocessor derivative used in audio_settings.h.

Returns

void

3.6.2.18 set_process_callback()

`void Handshake::set_process_callback ( )`
Function to set the callback function used for streaming the audio signal.

Returns

void

The documentation for this class was generated from the following files:

- include/handshake.h
- src/handshake.cpp

3.7 hardwareInterface Class Reference

The generic interface of the pi-loop. It has an input and an output interface.
`#include <interface.h>`

Public Member Functions

- hardwareInterface ()
  Class contructor.
- void get_metro_display (int)
  Method to call the metronome display function triggered by the Metronome.
- void get_display_initializer (DisplayInit)
  Method to initialize the display for some display components (i.e. if EFF1 is enabled by default on the current session)
- void listen (void(PiLoop::*_notify)(Trigger), PiLoop &)
  Prime method that runs on the ui thread.
- void display ()
  Prime method that runs on the display thread.
- void update (Response)
  Method to update the display thread upon user interventions.
- void deafen ()
  Method to shutdown the interface.

3.7.1 Detailed Description

The generic interface of the pi-loop. It has an input and an output interface.

3.7.2 Member Function Documentation

3.7.2.1 deafen()

`void hardwareInterface::deafen ( )`
Method to shutdown the interface.

Returns

void

3.7.2.2 display()

```
void hardwareInterface::display ( )
```

Prime method that runs on the display thread.

Returns

void

3.7.2.3 get_display_initializer()

```
void hardwareInterface::get_display_initializer (
            DisplayInit data )
```

Method to initialize the display for some display components (i.e. if EFF1 is enabled by default on the current session)

Parameters

int[9]	- integer data to initialize the display object.

Returns

void

3.7.2.4 get_metro_display()

```
void hardwareInterface::get_metro_display (
            int state )
```

Method to call the metronome display function triggered by the Metronome.

Parameters

int	- the metronome intonation signal.

Returns

void

3.7.2.5 listen()

```
void hardwareInterface::listen (
            void(PiLoop::*)(Trigger) _notify,
            PiLoop & pl )
```

Prime method that runs on the ui thread.

Parameters

void	(PiLoop::*_notify)(Trigger) - the function pointer fo the Piloop class that is responsible for receiving the trigger from the current function.
PiLoop&	- reference to the Piloop object.

Returns

void

3.7.2.6 update()

```
void hardwareInterface::update (
            Response response )
```

Method to update the display thread upon user interventions.

Parameters

Response	- post-process signal to update the display thread.

Returns

void

The documentation for this class was generated from the following files:

- include/interface.h
- src/interface.cpp

3.8 Keyboard Class Reference

The keyboard as the computer-based input interface.

```
#include <keyboard.h>
```

Public Member Functions

- Keyboard ()

  Class contructor.
- void is_pressed (Trigger &)

  Function to return if a keyboard input is pressed by the user.

3.8.1 Detailed Description

The keyboard as the computer-based input interface.

3.8.2 Member Function Documentation

3.8.2.1 is_pressed()

```
void Keyboard::is_pressed (
            Trigger & trigger )
```

Function to return if a keyboard input is pressed by the user.

Parameters

Trigger&	- the trigger to be returned.

Returns

void

The documentation for this class was generated from the following files:

- include/computer/keyboard.h
- src/computer/keyboard.cpp

3.9 Leds Class Reference

LEDs as the GPIO-based output interface.
`#include <leds.h>`

Public Member Functions

- void **initialize_leds** (int[8])
- void **display** ()
- void **perform_operation** (Trigger)
- void **tick_tock** (int)
- void **turnOff** ()

3.9.1 Detailed Description

LEDs as the GPIO-based output interface.
The documentation for this class was generated from the following files:

- include/gpio/leds.h
- src/gpio/leds.cpp

3.10 Looper Class Reference

The Looper class constisted of 3 loop-channels.
`#include <looper.h>`

Public Member Functions

- Looper ()

  Class contructor.
- void tap_alter_metronome (bool)

  Function to do the tap-tempo or to change the metronome to an alternative value of tempo and number of measures.
- LooperOutput * update_buffer (float *[F_NUM_INPUTS], bool[F_NUM_INPUTS])

  Method that receives the input buffer signal from the monitor.
- void recdub (int, bool, Response &)

  Function triggered by the REC button. Supports four (4) operations : record-dub/stop_recording/playback/erase_↩ previous Depending on the current state of the looper, it decides which one of them to apply.
- void stoperase (int, bool, Response &)

  Function triggered by the PLAY/STOP button. Supports three (3) operations : stop_playback/stop_recording/erase↩ _channel Depending on the current state of the looper, it decides which one of them to apply.
- void start_stop_all (bool, Response &)

  Function triggered by the START_ALL button. Supports two operations : start_stop_all/ erase_all Depending on the current state of the looper, it decides which one of them to apply.
- void volume_change (int, int)

  Sets a channel's volume with a specific value.
- void volume_up (int)

Increase the channel's volume by a channels-defined volume step. Used for when changing volumes with keyboard keys.

- void volume_down (int)

  Decrease the channel's volume by a channels-defined volume step. Used for when changing volumes with keyboard keys.

- bool save ()

  Save the looper state. Unsupported yet.

- void display_states ()

  Display the looper's current state. Used for debugging.

- int get_metronome_state ()

  Function used to update the metronome's display status.

- void reset ()

  Resets both the looper and the channels attached to it to the industrial conditions! (i.e. used for when changing session)

3.10.1 Detailed Description

The Looper class constisted of 3 loop-channels.

3.10.2 Member Function Documentation

3.10.2.1 display_states()

```
void Looper::display_states ( )
```

Display the looper's current state. Used for debugging.

Returns

void

3.10.2.2 get_metronome_state()

```
int Looper::get_metronome_state ( )
```

Function used to update the metronome's display status.

Returns

int - the intonation value of the metronome.

3.10.2.3 recdub()

```
void Looper::recdub (
            int channel,
            bool isHold,
            Response & response )
```

Function triggered by the REC button. Supports four (4) operations : record-dub/stop_recording/playback/erase_↩ previous Depending on the current state of the looper, it decides which one of them to apply.

Parameters

int	- the related channel
bool	- whether the button was pressed consecutively or not. If yes, it is an undub signal. Otherwise it is one among the rest of the operations.
Response&	- the output response that is used to update the display state of the looper.

Returns

void

3.10.2.4 reset()

```
void Looper::reset ( )
```

Resets both the looper and the channels attached to it to the industrial conditions! (i.e. used for when changing session)

Returns

void

3.10.2.5 save()

```
bool Looper::save ( )
```

Save the looper state. Unsupported yet.

Returns

void

3.10.2.6 start_stop_all()

```
void Looper::start_stop_all (
            bool isHold,
            Response & response )
```

Function triggered by the START_ALL button. Supports two operations : start_stop_all/ erase_all Depending on the current state of the looper, it decides which one of them to apply.

Parameters

bool	- whether the button was pressed consecutively or not. If yes, it is an erase-all-channels signal. Otherwise it is a signal that aims to pause all channels or to start playing all channels
Response&	- the output response that is used to update the display state of the looper.

Returns

void

3.10.2.7 stoperase()

```
void Looper::stoperase (
            int channel,
            bool isHold,
            Response & response )
```

Function triggered by the PLAY/STOP button. Supports three (3) operations : stop_playback/stop_↩ recording/erase_channel Depending on the current state of the looper, it decides which one of them to apply.

Parameters

int	- the related channel
bool	- whether the button was pressed consecutively or not. If yes, it is an erase signal. Otherwise it is one among the rest of the operations.

Parameters

Response&	- the output response that is used to update the display state of the looper.

Returns

void

3.10.2.8 tap_alter_metronome()

```
void Looper::tap_alter_metronome (
            bool isHold )
```

Function to do the tap-tempo or to change the metronome to an alternative value of tempo and number of measures.

Parameters

bool	- if the button is pressed consecutively. if true, then it alters the tempo. Otherwise it is a tap-tempo signal.

Returns

void

3.10.2.9 update_buffer()

```
LooperOutput * Looper::update_buffer (
            float * input[F_NUM_INPUTS],
            bool armEnabled[F_NUM_INPUTS] )
```

Method that receives the input buffer signal from the monitor.

Parameters

float∗*[F_NUM_INPU*↩*TS]*	- the input buffer sent by the monitor.
bool[F_NUM_INPUTS]	- the arm states

Returns

std::array< std::array<float, BUFFER_SIZE>, F_NUM_OUTPUTS> - the looper ouput buffer

3.10.2.10 volume_change()

```
void Looper::volume_change (
            int channel,
            int volume )
```

Sets a channel's volume with a specific value.

Parameters

int	- the corresponding channel
int	- the value of the volume as type of integer percentage.

Returns

void

3.10.2.11 volume_down()

```
void Looper::volume_down (
            int channel )
```

Decrease the channel's volume by a channels-defined volume step. Used for when changing volumes with keyboard keys.

Parameters

int	- the corresponding channel

Returns

void

3.10.2.12 volume_up()

```
void Looper::volume_up (
            int channel )
```

Increase the channel's volume by a channels-defined volume step. Used for when changing volumes with keyboard keys.

Parameters

int	- the corresponding channel

Returns

void

The documentation for this class was generated from the following files:

- include/looper.h
- src/looper.cpp

3.11 LooperState Struct Reference

Stores the looper state.

`#include <response.h>`

Public Member Functions

- LooperState ()

  Default constructor.
- LooperState (const LooperState &obj)

  Copy constructor.
- LooperState & operator= (const LooperState &obj)

  Assignment operator.
- void reset ()

  Reset function resets the object member variables to the default values.

Public Attributes

- std::atomic< bool > **playbacks** [3]
- std::atomic< bool > **record** [3]
- std::atomic< int > **num_tracks** [3]
- std::atomic< int > **is_changed**

3.11.1 Detailed Description

Stores the looper state.
The documentation for this struct was generated from the following file:

- include/response.h

3.12 Menu Class Reference

The programs menu that acts as the session manager.
`#include <menu.h>`

Public Member Functions

- Menu ()

  Class constuctor.
- void setup ()

  Function to apply setup actions after initialization and before running.
- void load_session ()

  Prime function to run on the session thread.
- void notify_menu (Trigger, Response &)

  Function to receive a trigger message and return a response message .
- void unload ()

  Function to shutdown menu.
- void set_display_initializer (std::function< void(DisplayInit)>)

  Function that carries a function pointer to the session obj.
- void set_metro_display (std::function< void(int)>)

  Function that carries a function pointer to the session obj.

3.12.1 Detailed Description

The programs menu that acts as the session manager.

3.12.2 Member Function Documentation

3.12.2.1 load_session()

`void Menu::load_session ( )`
Prime function to run on the session thread.

Returns

void

3.12.2.2 notify_menu()

```
void Menu::notify_menu (
            Trigger trigger,
            Response & response )
```

Function to receive a trigger message and return a response message .

Parameters

Trigger	- the user input signal
Response&	- the response signal (passed by reference) to reach the display methods.

Returns

void

3.12.2.3 set_display_initializer()

```
void Menu::set_display_initializer (
            std::function< void(DisplayInit)> f )
```

Function that carries a function pointer to the session obj.

Parameters

std::function<void(int[9])>	- the function to initialize the display object with info obtained by the session object.

Returns

void

3.12.2.4 set_metro_display()

```
void Menu::set_metro_display (
            std::function< void(int)> f )
```

Function that carries a function pointer to the session obj.

Parameters

std::function<void(int)>	- the function that is going to be called each time the metronomes ticks.

Returns

void

3.12.2.5 setup()

```
void Menu::setup ( )
```

Function to apply setup actions after initialization and before running.

Returns

void

3.12.2.6 unload()

```
void Menu::unload ( )
```

Function to shutdown menu.

Returns

void

The documentation for this class was generated from the following files:

- include/menu.h
- src/menu.cpp

3.13 Metronome Class Reference

A metronome defined within Looper.

`#include <metronome.h>`

Public Member Functions

- Metronome ()

  Class contructor.
- void lock ()

  function to set lock to true, so as to prevent metronome from changing.
- void unlock ()

  function to set lock to false, so as to enable resetting of the metronome.
- void pause ()

  function to pause the counting of the metronome.
- void unpause ()

  function to unpause the metronome.
- void start_timing ()

  function to start timing the first loop.
- void stop_timing (int)

  function to stop timing and set the tempo.
- void tap_tempo ()

  function to read taps from the user for setting the tempo
- int tick_tock ()

  function that counts the time and does the tick-tocking
- void alter_tempo ()

  function that changes the tempo to the next most suitable form.
- void clear ()

  Function used to reset the metronome.(i.e. for when changing session)
- void sync ()

  Helper function to synchronize metronome with the looper.

3.13.1 Detailed Description

A metronome defined within Looper.

3.13.2 Member Function Documentation

3.13.2.1 alter_tempo()

```
void Metronome::alter_tempo ( )
```

function that changes the tempo to the next most suitable form.

Returns

void

3.13.2.2 clear()

```
void Metronome::clear ( )
```

Function used to reset the metronome.(i.e. for when changing session)

Returns

void

3.13.2.3 lock()

```
void Metronome::lock ( )
```

function to set lock to true, so as to prevent metronome from changing.

Returns

void

3.13.2.4 pause()

```
void Metronome::pause ( )
```

function to pause the counting of the metronome.

Returns

void

3.13.2.5 start_timing()

```
void Metronome::start_timing ( )
```

function to start timing the first loop.

Returns

void

3.13.2.6 stop_timing()

```
void Metronome::stop_timing (
            int loop_length )
```

function to stop timing and set the tempo.

Parameters

int	- the loop length in samples.

Returns

void

3.13.2.7 sync()

`void Metronome::sync ( )`
Helper function to synchronize metronome with the looper.

Returns

void

3.13.2.8 tap_tempo()

`void Metronome::tap_tempo ( )`
function to read taps from the user for setting the tempo

Returns

void

3.13.2.9 tick_tock()

`int Metronome::tick_tock ( )`
function that counts the time and does the tick-tocking

Returns

int - the intontation of the tick. 0 if the first of the loop, 1 if first of the measure, 2 otherwise

3.13.2.10 unlock()

`void Metronome::unlock ( )`
function to set lock to false, so as to enable resetting of the metronome.

Returns

void

3.13.2.11 unpause()

`void Metronome::unpause ( )`
function to unpause the metronome.

Returns

void

The documentation for this class was generated from the following files:

- include/metronome.h
- src/metronome.cpp

3.14 Mixer Class Reference

The mixer module responsible to stream the output signal.
`#include <mixer.h>`

Public Member Functions

- Mixer ()

  Class contructor.
- void update_buffer (float *[F_NUM_INPUTS], float *[F_NUM_OUTPUTS], LooperOutput &, bool[F_NUM_↩ INPUTS])

  Function that streams the output audio signal.
- void save_jam (std::string)

  Function that saves the last minutes to a file.

3.14.1 Detailed Description

The mixer module responsible to stream the output signal.

3.14.2 Member Function Documentation

3.14.2.1 save_jam()

```
void Mixer::save_jam (
            std::string savepath )
```

Function that saves the last minutes to a file.

Parameters

std::string	- the path to save the file.

Returns

void

3.14.2.2 update_buffer()

```
void Mixer::update_buffer (
            float * input_buffers[F_NUM_INPUTS],
            float * output_buffers[F_NUM_OUTPUTS],
            LooperOutput & looper_buffers,
            bool monitorIn[F_NUM_INPUTS] )
```

Function that streams the output audio signal.

Parameters

float[F_NUM_INPUTS]*	- the monitor audio input signal
float[F_NUM_OUTPU↩ TS]*	- the output buffers connected to the speakers
LooperOutput	- the looper output buffers
bool[F_NUM_INPUTS]	- bool array to indicate if monitorIn is enabled.

Returns

void

The documentation for this class was generated from the following files:

- include/mixer.h
- src/mixer.cpp

3.15 Monitor Class Reference

The monitor module encapsulates the jack audio client.
`#include <monitor.h>`

Public Member Functions

- Monitor ()

  Class contructor.
- void connect (char *)

  Function to connect the audio client to the audio server.
- void disconnect ()

  Function to disconnect the jack audio client from the audio server.
- void initialize_effects (const bool[F_NUM_INPUTS][NUM_EFFECTS])

  Function to initialize the effects (enabled/disabled) depending on the state of the session when last saved.
- bool toggle_effect (int, int)

  Function to change state of effect.
- void mute_input (int)

  Function to mute the input signal.
- void mute_output (int)

  Function to mute the output signal.
- int process ()

  The callback function used to retrieve the input buffers.
- void set_stream_buffer (std::function< void(float *[F_NUM_INPUTS], float *[F_NUM_OUTPUTS])>)

  Function to set a member function with a function pointer to a non member function from another class so as to have the incomming buffers updated outside of the class scope.
- bool toggle_states (bool, int)

  Function to change the state of the monitor options (ARM/MNTR)
- void update_states (bool, int, bool)

  Function to set the state of the monitor options (ARM/MNTR) with a specific value. Used for initialization.
- void get_states (bool[F_NUM_INPUTS], bool)

  Function to get the current state of the monitor options (ARM/MNTR).

3.15.1 Detailed Description

The monitor module encapsulates the jack audio client.

3.15.2 Member Function Documentation

3.15.2.1 connect()

```
void Monitor::connect (
            char * name )
```

Function to connect the audio client to the audio server.

Parameters

char∗	- the client name

Returns

void

3.15.2.2 disconnect()

```
void Monitor::disconnect ( )
```

Function to disconnect the jack audio client from the audio server.

Returns

void

3.15.2.3 get_states()

```
void Monitor::get_states (
            bool states[F_NUM_INPUTS],
            bool is_mntr )
```

Function to get the current state of the monitor options (ARM/MNTR).

Parameters

bool[F_NUM_INPU↩ TS]	- the values to be set.
bool	- if it is MNTR or ARM.

Returns

void

3.15.2.4 initialize_effects()

```
void Monitor::initialize_effects (
            const bool effects_curr_state[F_NUM_INPUTS][NUM_EFFECTS] )
```

Function to initialize the effects (enabled/disabled) depending on the state of the session when last saved.

Parameters

const	bool[F_NUM_INPUTS][NUM_EFFECTS] - a table of boolean to declare the status (enabled/disabled) of each effect for each input channel

Returns

void

3.15.2.5 mute_input()

```
void Monitor::mute_input (
            int device )
```

Function to mute the input signal.

Parameters

int	- refers to the corresponding channel

Returns

void

3.15.2.6 mute_output()

```
void Monitor::mute_output (
            int device )
```

Function to mute the output signal.

Parameters

int	- refers to the corresponding output channel

Returns

void

3.15.2.7 process()

```
int Monitor::process ( )
```

The callback function used to retrieve the input buffers.

Returns

int - zero on success, non-zero on error

The process callback for this JACK application is called in a special realtime thread once for each audio cycle.

3.15.2.8 set_stream_buffer()

```
void Monitor::set_stream_buffer (
            std::function< void(float *[F_NUM_INPUTS], float *[F_NUM_OUTPUTS])> )
```

Function to set a member function with a function pointer to a non member function from another class so as to have the incomming buffers updated outside of the class scope.

Parameters

std::function<void(float	*[F_NUM_INPUTS],float *[F_NUM_OUTPUTS])> - the pointer to the non-member function.

Returns

void

3.15.2.9 toggle_effect()

```
bool Monitor::toggle_effect (
```

```
            int ch,
            int eff )
```

Function to change state of effect.

Parameters

int	- refers to the corresponding channel
int	- refers to the corresponding effect

Returns

void

3.15.2.10 toggle_states()

```
bool Monitor::toggle_states (
            bool is_mntr,
            int channel )
```

Function to change the state of the monitor options (ARM/MNTR)

Parameters

bool	- if it is MNTR or ARM.
int	- the corresponding channel

Returns

bool - the current state of the monitor-button.

3.15.2.11 update_states()

```
void Monitor::update_states (
            bool is_mntr,
            int input_channel,
            bool val )
```

Function to set the state of the monitor options (ARM/MNTR) with a specific value. Used for initialization.

Parameters

bool	- if it is MNTR or ARM.
int	- the corresponding channel
bool	- the value to be set.

Returns

void

The documentation for this class was generated from the following files:

- include/monitor.h
- src/monitor.cpp

3.16 PiLoop Class Reference

The threading manager. It starts the program.

`#include <piloop.h>`

Public Member Functions

- PiLoop ()

  Class contructor. Responsible for setting up the threads and some function pointers to callback functions.
- void start_console ()

  Runs all the four threads.

3.16.1 Detailed Description

The threading manager. It starts the program.

3.16.2 Member Function Documentation

3.16.2.1 start_console()

`void PiLoop::start_console ( )`
Runs all the four threads.

Returns

void

The documentation for this class was generated from the following files:

- include/piloop.h
- src/piloop.cpp

3.17 Potentiometers Class Reference

Potentiometers to adjust the looper-channel's volume.
`#include <potentiometers.h>`

Public Member Functions

- void **setup** (int)
- void **is_changed** (Trigger &)

3.17.1 Detailed Description

Potentiometers to adjust the looper-channel's volume.
The documentation for this class was generated from the following files:

- include/gpio/potentiometers.h
- src/gpio/potentiometers.cpp

3.18 Response Struct Reference

Defines a response message destinated to the displayer module of the program.
`#include <response.h>`

Public Member Functions

- Response ()

  Default constructor.
- Response (const Response &obj)

  Copy constructor.
- Response & operator= (const Response &obj)

  Assignment operator.
- bool isEmpty ()

  Function that returns if a Response message is empty.
- void reset ()

  Reset function resets the object member variables to the default values.

Public Attributes

- std::atomic< int > **holder** {-1}
- std::atomic< Message > **msg** {EMPTY}
- std::atomic< int > **value** {-1}
- LooperState **looper_state**

3.18.1 Detailed Description

Defines a response message destinated to the displayer module of the program.

3.18.2 Member Function Documentation

3.18.2.1 isEmpty()

```
bool Response::isEmpty ( )  [inline]
```

Function that returns if a Response message is empty.

Returns

bool - True if is empty (if msg == EMPTY or if looper_state is not changed). Otherwise false

3.18.2.2 reset()

```
void Response::reset ( )  [inline]
```

Reset function resets the object member variables to the default values.

Returns

void

The documentation for this struct was generated from the following file:

- include/response.h

3.19 Screen Class Reference

The screen as the computer-based output interface.

`#include <screen.h>`

Public Member Functions

- Screen ()

  Class contructor.
- void initialize_states (DisplayInit)

  Method to impicitly initialize some output display components.
- void display ()

  Threaded function running on the display thread.
- void perform_operation (Response)

  Function running on the session thread, and carries the system's response to the displayer.
- void tick_tock (int)

  Method to display the metronome's state.
- void turnOff ()

  Method to shutdown the displayer.

3.19.1 Detailed Description

The screen as the computer-based output interface.

3.19.2 Member Function Documentation

3.19.2.1 display()

```
void Screen::display ( )
```

Threaded function running on the display thread.

Returns

void

3.19.2.2 initialize_states()

```
void Screen::initialize_states (
            DisplayInit comp_states )
```

Method to impicitly initialize some output display components.

Parameters

DisplayInit	- effects, monitor states + the current session)

Returns

void

3.19.2.3 perform_operation()

```
void Screen::perform_operation (
            Response response )
```

Function running on the session thread, and carries the system's response to the displayer.

Returns

void

3.19.2.4 tick_tock()

```
void Screen::tick_tock (
            int intonation )
```

Method to display the metronome's state.

Parameters

int	- the metronome's intonation signal

Returns

void

3.19.2.5 turnOff()

```
void Screen::turnOff ( )
```

Method to shutdown the displayer.

Returns

void

The documentation for this class was generated from the following files:

- include/computer/screen.h
- src/computer/screen.cpp

3.20 Session Class Reference

Defines a musical session. It is consisted of a Monitor, a Looper and a Mixer.
`#include <session.h>`

Public Member Functions

- Session ()

  Class contructor.
- void setup ()

  Setup whatever needed for implicit initiatialization of the imported classes.
- void load ()

  Threaded function. This one is to have the audio client up and running.
- void save ()

  Function to save the current session. Currently it is partially unsupported (Looper session will not be saved).
- void migrate (int)

  Function used to change session. @params int - the next session.
- void evacuate ()

  Function shutdown the session.
- void notify_session (Trigger, Response &)

  Function used to retrieve the user's input trigger, and process the message and pass it all around the program, getting sure that a response will be returned for display.
- void save_jam ()

  Function to save the last minutes of the current jam.
- void set_name (const char *)

  Function to set the name of the current session.
- const char * get_name ()

Function to return the name of the current session.

- void set_metronome_display (std::function< void(int)>)

  Function to pass the metronome-display-function-pointer.

- void set_disp_initializer (std::function< void(DisplayInit)>)

  Function that gets a function pointer to set a member function, which is used to implicitly initialize the display module with default/predefined values.

3.20.1 Detailed Description

Defines a musical session. It is consisted of a Monitor, a Looper and a Mixer.

3.20.2 Member Function Documentation

3.20.2.1 evacuate()

```
void Session::evacuate ( )
```

Function shutdown the session.

Returns

void

3.20.2.2 get_name()

```
const char * Session::get_name ( )
```

Function to return the name of the current session.

Returns

const char* - the name of the current session.

3.20.2.3 load()

```
void Session::load ( )
```

Threaded function. This one is to have the audio client up and running.

Returns

void

3.20.2.4 migrate()

```
void Session::migrate (
            int next_session )
```

Function used to change session. @params int - the next session.

Returns

void

3.20.2.5 notify_session()

```
void Session::notify_session (
            Trigger trigger,
            Response & response )
```

Function used to retrieve the user's input trigger, and process the message and pass it all around the program, getting sure that a response will be returned for display.

Returns

void

3.20.2.6 save()

```
void Session::save ( )
```

Function to save the current session. Currently it is partially unsupported (Looper session will not be saved).

Returns

void

3.20.2.7 save_jam()

```
void Session::save_jam ( )
```

Function to save the last minutes of the current jam.

Returns

void

3.20.2.8 set_disp_initializer()

```
void Session::set_disp_initializer (
            std::function< void(DisplayInit)> di )
```

Function that gets a function pointer to set a member function, which is used to implicitly initialize the display module with default/predefined values.

Parameters

std::function<void(int[9])>	- the pointer to the function that takes 9 integer values to initialize the output states before use.

Returns

void

3.20.2.9 set_metronome_display()

```
void Session::set_metronome_display (
            std::function< void(int)> metrodisp )
```

Function to pass the metronome-display-function-pointer.

Parameters

std::function<void(int)>	- the metronome display function

Returns

void

3.20.2.10 set_name()

```
void Session::set_name (
            const char * name )
```

Function to set the name of the current session.

Parameters

const	char* - the new name

Returns

void

3.20.2.11 setup()

```
void Session::setup ( )
```

Setup whatever needed for implicit initiatialization of the imported classes.

Returns

void

The documentation for this class was generated from the following files:

- include/session.h
- src/session.cpp

3.21 time_signature Struct Reference

Struct to hold the information of rythm.

`#include <metronome.h>`

Public Attributes

- int **numerator**
- int **denominator**

3.21.1 Detailed Description

Struct to hold the information of rythm.

Parameters

int	numerator - the rythm numerator
int	denominator - the rythm denominator

The documentation for this struct was generated from the following file:

- include/metronome.h

3.22 Trigger Struct Reference

Holds the triggers of the user.
`#include <trigger.h>`

Public Member Functions

- Trigger ()

  Default constructor.
- Trigger (const Trigger &obj)

  Copy constructor.
- Trigger & operator= (const Trigger &obj)

  Assignment operator.
- void reset ()

  Reset function resets the object member variables to the default values.

Public Attributes

- std::atomic< Control > **control** {ISEMPTY}
- std::atomic< int > **subval** {-1}

3.22.1 Detailed Description

Holds the triggers of the user.

3.22.2 Member Function Documentation

3.22.2.1 reset()

`void Trigger::reset ( ) [inline]`
Reset function resets the object member variables to the default values.

Returns

void

The documentation for this struct was generated from the following file:

- include/trigger.h

3.23 UI Class Reference

Defines a computer based interface constisted of a keyboard and the screen.
`#include <computer.h>`

Public Member Functions

- UI ()

  Class contructor.
- void listen_user (Trigger &)

  Method to pass the trigger stucture by reference to a function that gets the keyboard input messages by the user.
- void update_output_state (Response)

  Method to update the output based on the system's response message.
- void show ()

  Method that runs on the display thread.

- void initialize_display (DisplayInit)
- void metro_display (int)

  Method to call the displayer's tick_tock function.
- void turnOff ()

  Method to shutdown UI.
- void **listen_user** (Trigger &)
- void **update_output_state** (Response)
- void **show** ()
- void **initialize_display** (int[8])
- void **metro_display** (int)
- void **turnOff** ()

3.23.1 Detailed Description

Defines a computer based interface constisted of a keyboard and the screen.
Defines a GPIO based interface constisted of buttons, potentiometers and leds.

3.23.2 Member Function Documentation

3.23.2.1 initialize_display()

```
void UI::initialize_display (
            DisplayInit data )
```

Method to pass some initialization values to the displayer before the program starts

Parameters

DisplayInit	- integer data to initialize the display object.

Returns

void

3.23.2.2 listen_user()

```
void UI::listen_user (
            Trigger & trigger )
```

Method to pass the trigger stucture by reference to a function that gets the keyboard input messages by the user.

Parameters

Trigger&	- The trigger to be set by the user using the pc-keyboard.

Returns

void

3.23.2.3 metro_display()

```
void UI::metro_display (
            int state )
```

Method to call the displayer's tick_tock function.

Parameters

int	- the intonation signal obtained from the metronome obj.

Returns

void

3.23.2.4 show()

```
void UI::show ( )
```

Method that runs on the display thread.

Returns

void

3.23.2.5 turnOff()

```
void UI::turnOff ( )
```

Method to shutdown UI.

Returns

void

3.23.2.6 update_output_state()

```
void UI::update_output_state (
            Response response )
```

Method to update the output based on the system's response message.

Parameters

Response	- the program's response carried to the displayer.

Returns

void

The documentation for this class was generated from the following files:

- include/computer.h
- include/gpio.h
- src/computer.cpp
- src/gpio.cpp

Index

activate
    Handshake, 17
add_track
    Channel, 9
alter_tempo
    Metronome, 31
apply
    Effects, 15
AudioServer, 7
    start, 7
    stop, 7

Buttons, 8

Channel, 8
    add_track, 9
    clean, 9
    dub, 9
    get_name, 9
    get_num_tracks, 9
    get_out_signal, 10
    get_volume, 10
    isEmpty, 10
    rec, 10
    reset, 10
    reset_num_tracks, 11
    set_name, 11
    set_volume, 11
    undub, 11
    update_rec_signal, 11
    volume_down, 12
    volume_up, 12
check_status
    Handshake, 17
clean
    Channel, 9
clear
    Metronome, 32
Config, 12
    display, 13
    get_button_state, 13
    get_curr_session, 13
    get_max_sessions, 13
    open, 14
    reset, 14
    save, 14
connect
    Monitor, 35
connect_input_device
    Handshake, 17
connect_output_device
    Handshake, 17

deafen
    hardwareInterface, 21
disconnect
    Monitor, 36
disconnect_client
    Handshake, 17
disconnect_input_device
    Handshake, 18
disconnect_output_device
    Handshake, 18
display
    Config, 13
    hardwareInterface, 21
    Screen, 41
display_states
    Looper, 25
dub
    Channel, 9

Effects, 14
    apply, 15
    initialize_effects, 15
    toggle_effect, 15
evacuate
    Session, 43

get_button_state
    Config, 13
get_curr_session
    Config, 13
get_display_initializer
    hardwareInterface, 22
get_input_buffer
    Handshake, 18
get_max_sessions
    Config, 13
get_metro_display
    hardwareInterface, 22
get_metronome_state
    Looper, 25
get_name
    Channel, 9
    Session, 43
get_num_tracks
    Channel, 9
get_out_signal
    Channel, 10

get_output_buffer
Handshake, 19
get_states
Monitor, 36
get_volume
Channel, 10

Handshake, 16
activate, 17
check_status, 17
connect_input_device, 17
connect_output_device, 17
disconnect_client, 17
disconnect_input_device, 18
disconnect_output_device, 18
get_input_buffer, 18
get_output_buffer, 19
link_client, 19
prevent_failure, 20
register_devices, 20
register_input_port, 20
register_output_port, 20
set_buffer_size, 20
set_process_callback, 21
hardwareInterface, 21
deafen, 21
display, 21
get_display_initializer, 22
get_metro_display, 22
listen, 22
update, 23

initialize_display
UI, 47
initialize_effects
Effects, 15
Monitor, 36
initialize_states
Screen, 41
is_pressed
Keyboard, 23
isEmpty
Channel, 10
Response, 40

Keyboard, 23
is_pressed, 23

Leds, 24
link_client
Handshake, 19
listen
hardwareInterface, 22
listen_user
UI, 47
load
Session, 43
load_session
Menu, 29

lock
Metronome, 32
Looper, 24
display_states, 25
get_metronome_state, 25
recdub, 25
reset, 26
save, 26
start_stop_all, 26
stoperase, 26
tap_alter_metronome, 27
update_buffer, 27
volume_change, 27
volume_down, 28
volume_up, 28
LooperState, 28

Menu, 29
load_session, 29
notify_menu, 29
set_display_initializer, 30
set_metro_display, 30
setup, 30
unload, 30
metro_display
UI, 47
Metronome, 31
alter_tempo, 31
clear, 32
lock, 32
pause, 32
start_timing, 32
stop_timing, 32
sync, 33
tap_tempo, 33
tick_tock, 33
unlock, 33
unpause, 33
migrate
Session, 43
Mixer, 34
save_jam, 34
update_buffer, 34
Monitor, 35
connect, 35
disconnect, 36
get_states, 36
initialize_effects, 36
mute_input, 36
mute_output, 37
process, 37
set_stream_buffer, 37
toggle_effect, 37
toggle_states, 38
update_states, 38
mute_input
Monitor, 36
mute_output
Monitor, 37

notify_menu
Menu, 29
notify_session
Session, 43

open
Config, 14

pause
Metronome, 32
perform_operation
Screen, 41
PiLoop, 38
start_console, 39
Potentiometers, 39
prevent_failure
Handshake, 20
process
Monitor, 37

rec
Channel, 10
recdub
Looper, 25
register_devices
Handshake, 20
register_input_port
Handshake, 20
register_output_port
Handshake, 20
reset
Channel, 10
Config, 14
Looper, 26
Response, 40
Trigger, 46
reset_num_tracks
Channel, 11
Response, 39
isEmpty, 40
reset, 40

save
Config, 14
Looper, 26
Session, 44
save_jam
Mixer, 34
Session, 44
Screen, 40
display, 41
initialize_states, 41
perform_operation, 41
tick_tock, 41
turnOff, 42
Session, 42
evacuate, 43
get_name, 43
load, 43
migrate, 43
notify_session, 43
save, 44
save_jam, 44
set_disp_initializer, 44
set_metronome_display, 44
set_name, 45
setup, 45
set_buffer_size
Handshake, 20
set_disp_initializer
Session, 44
set_display_initializer
Menu, 30
set_metro_display
Menu, 30
set_metronome_display
Session, 44
set_name
Channel, 11
Session, 45
set_process_callback
Handshake, 21
set_stream_buffer
Monitor, 37
set_volume
Channel, 11
setup
Menu, 30
Session, 45
show
UI, 48
start
AudioServer, 7
start_console
PiLoop, 39
start_stop_all
Looper, 26
start_timing
Metronome, 32
stop
AudioServer, 7
stop_timing
Metronome, 32
stoperase
Looper, 26
sync
Metronome, 33

tap_alter_metronome
Looper, 27
tap_tempo
Metronome, 33
tick_tock
Metronome, 33
Screen, 41
time_signature, 45
toggle_effect
Effects, 15

Monitor, 37
toggle_states
Monitor, 38
Trigger, 46
reset, 46
turnOff
Screen, 42
UI, 48

UI, 46
initialize_display, 47
listen_user, 47
metro_display, 47
show, 48
turnOff, 48
update_output_state, 48
undub
Channel, 11
unload
Menu, 30
unlock
Metronome, 33
unpause
Metronome, 33
update
hardwareInterface, 23
update_buffer
Looper, 27
Mixer, 34
update_output_state
UI, 48
update_rec_signal
Channel, 11
update_states
Monitor, 38

volume_change
Looper, 27
volume_down
Channel, 12
Looper, 28
volume_up
Channel, 12
Looper, 28