Sequencer64 Developer's Reference Manual

0.9.9.5

Generated by Doxygen 1.8.9.1

Mon Oct 12 2015 06:27:30

Contents

1 Sequencer64 **1**

1.1 Introduction 1

2 User Testing of Sequencer64 with Yoshimi **3**

2.1 Introduction 3

2.2 Smoke Test 3

2.3 Tests in the Patterns Window 4

2.3.1 Patterns Window Key Shortcuts 4

2.3.2 The Sequencer64 User File 4

3 Licenses **5**

3.1 License Terms for the This Project. 5

3.2 XPC Application License 5

3.3 XPC Library License 6

3.4 XPC Documentation License 6

3.5 XPC Affero License 6

3.6 XPC License Summary 7

4 Todo List **9**

5 Hierarchical Index **11**

5.1 Class Hierarchy 11

6 Data Structure Index **13**

6.1 Data Structures 13

7 Data Structure Documentation **15**

7.1 seq64::automutex Class Reference 15

7.1.1 Detailed Description 15

7.2 seq64::click Class Reference 15

7.2.1 Detailed Description 16

7.2.2 Constructor & Destructor Documentation 16

7.2.2.1 click 16

7.2.2.2 click 16

7.2.2.3 click 16
7.2.3 Member Function Documentation 16
7.2.3.1 operator= 16
7.3 seq64::condition_var Class Reference 17
7.3.1 Detailed Description 17
7.4 seq64::configfile Class Reference 18
7.4.1 Constructor & Destructor Documentation 19
7.4.1.1 configfile 19
7.4.2 Member Function Documentation 19
7.4.2.1 next_data_line 19
7.4.2.2 line_after 19
7.4.3 Field Documentation 19
7.4.3.1 m_line 19
7.5 seq64::event Class Reference 19
7.5.1 Detailed Description 21
7.5.2 Member Function Documentation 22
7.5.2.1 operator< 22
7.5.2.2 mod_timestamp 22
7.5.2.3 set_status 22
7.5.2.4 set_data 22
7.5.2.5 set_data 22
7.5.2.6 get_data 23
7.5.2.7 append_sysex 23
7.5.2.8 get_rank 23
7.6 seq64::event_list Class Reference 23
7.6.1 Detailed Description 24
7.6.2 Constructor & Destructor Documentation 24
7.6.2.1 event_list 24
7.6.3 Member Function Documentation 24
7.6.3.1 operator= 24
7.6.3.2 count 25
7.6.3.3 add 25
7.7 seq64::gui_assistant Class Reference 25
7.7.1 Detailed Description 25
7.7.2 Constructor & Destructor Documentation 25
7.7.2.1 gui_assistant 25
7.8 seq64::gui_play_base Class Reference 26
7.8.1 Detailed Description 26
7.9 seq64::jack_assistant Class Reference 26
7.9.1 Constructor & Destructor Documentation 27

7.9.1.1 jack_assistant 27
7.9.2 Member Function Documentation 27
7.9.2.1 init 27
7.9.2.2 position 27
7.9.2.3 output 27
7.9.3 Friends And Related Function Documentation 28
7.9.3.1 jack_sync_callback 28
7.9.3.2 jack_shutdown 28
7.9.3.3 jack_timebase_callback 28
7.10 seq64::jack_scratchpad Struct Reference 28
7.10.1 Detailed Description 28
7.11 seq64::keys_perform Class Reference 28
7.11.1 Detailed Description 29
7.11.2 Constructor & Destructor Documentation 29
7.11.2.1 ∼keys_perform 29
7.11.3 Member Function Documentation 29
7.11.3.1 set_keys 29
7.11.3.2 get_keys 30
7.11.3.3 show_ui_sequence_key 30
7.11.3.4 key_name 30
7.11.3.5 set_all_key_events 30
7.11.3.6 set_all_key_groups 30
7.11.3.7 set_key_event 30
7.11.3.8 set_key_group 30
7.12 seq64::keys_perform_transfer Struct Reference 31
7.13 seq64::keystroke Class Reference 31
7.13.1 Detailed Description 31
7.13.2 Constructor & Destructor Documentation 31
7.13.2.1 keystroke 31
7.13.2.2 keystroke 32
7.13.3 Member Function Documentation 32
7.13.3.1 operator= 32
7.13.3.2 is_letter 32
7.14 seq64::lash Class Reference 32
7.14.1 Detailed Description 33
7.14.2 Constructor & Destructor Documentation 33
7.14.2.1 lash 33
7.14.3 Member Function Documentation 33
7.14.3.1 set_alsa_client_id 33
7.15 seq64::midi_container Class Reference 33

7.15.1 Member Function Documentation . . . 35
7.15.1.1 fill . . . 35
7.15.1.2 put . . . 35
7.15.1.3 get . . . 35
7.15.1.4 position . . . 35
7.16 seq64::midi_list Class Reference . . . 35
7.16.1 Member Function Documentation . . . 36
7.16.1.1 put . . . 37
7.16.1.2 get . . . 37
7.17 seq64::midi_vector Class Reference . . . 37
7.17.1 Member Function Documentation . . . 38
7.17.1.1 put . . . 38
7.17.1.2 get . . . 38
7.18 seq64::midifile Class Reference . . . 38
7.18.1 Detailed Description . . . 38
7.18.2 Constructor & Destructor Documentation . . . 38
7.18.2.1 midifile . . . 38
7.18.3 Member Function Documentation . . . 39
7.18.3.1 parse . . . 39
7.19 seq64::mutex Class Reference . . . 39
7.20 seq64::optionsfile Class Reference . . . 40
7.20.1 Detailed Description . . . 41
7.20.2 Member Function Documentation . . . 41
7.20.2.1 parse . . . 41
7.20.2.2 write . . . 42
7.21 seq64::perform Class Reference . . . 42
7.21.1 Detailed Description . . . 46
7.21.2 Constructor & Destructor Documentation . . . 46
7.21.2.1 perform . . . 46
7.21.2.2 ∼perform . . . 47
7.21.3 Member Function Documentation . . . 47
7.21.3.1 init . . . 47
7.21.3.2 launch_input_thread . . . 47
7.21.3.3 launch_output_thread . . . 47
7.21.3.4 init_jack . . . 47
7.21.3.5 add_sequence . . . 47
7.21.3.6 clear_sequence_triggers . . . 47
7.21.3.7 is_sequence_valid . . . 48
7.21.3.8 is_sequence_invalid . . . 48
7.21.3.9 move_triggers . . . 48

7.21.3.10 copy_triggers 48
7.21.3.11 get_midi_control_toggle 48
7.21.3.12 get_midi_control_on 48
7.21.3.13 get_midi_control_off 48
7.21.3.14 get_screen_set_notepad 48
7.21.3.15 set_screen_set_notepad 49
7.21.3.16 set_screenset 49
7.21.3.17 set_playing_screenset 49
7.21.3.18 unset_mode_group_learn 49
7.21.3.19 select_mute_group 49
7.21.3.20 start 49
7.21.3.21 stop 49
7.21.3.22 position_jack 50
7.21.3.23 all_notes_off 50
7.21.3.24 set_was_active 50
7.21.3.25 is_active 50
7.21.3.26 is_dirty_main 50
7.21.3.27 is_dirty_edit 50
7.21.3.28 is_dirty_perf 50
7.21.3.29 is_dirty_names 51
7.21.3.30 new_sequence 51
7.21.3.31 reset_sequences 51
7.21.3.32 play 51
7.21.3.33 set_orig_ticks 51
7.21.3.34 set_bpm 51
7.21.3.35 set_sequence_control_status 52
7.21.3.36 unset_sequence_control_status 52
7.21.3.37 output_func 52
7.21.3.38 get_max_trigger 52
7.21.3.39 set_offset 52
7.21.3.40 show_ui_sequence_key 52
7.21.3.41 start_playing 52
7.21.3.42 decrement_bpm 52
7.21.3.43 increment_bpm 53
7.21.3.44 set_input_bus 53
7.21.3.45 mainwnd_key_event 53
7.21.3.46 perfroll_key_event 53
7.22 seq64::performcallback Struct Reference 53
7.22.1 Detailed Description 53
7.23 rc_settings Class Reference 53

7.23.1 Member Function Documentation 55
7.23.1.1 home_config_directory 55
7.24 seq64::sequence Class Reference 55
7.24.1 Detailed Description 60
7.24.2 Member Enumeration Documentation 60
7.24.2.1 select_action_e 60
7.24.3 Member Function Documentation 61
7.24.3.1 operator= 61
7.24.3.2 event_count 61
7.24.3.3 push_undo 61
7.24.3.4 pop_undo 61
7.24.3.5 pop_redo 61
7.24.3.6 push_trigger_undo 61
7.24.3.7 set_bpm 61
7.24.3.8 set_bw 61
7.24.3.9 get_bw 61
7.24.3.10 set_rec_vol 61
7.24.3.11 set_length 61
7.24.3.12 set_playing 62
7.24.3.13 toggle_queued 63
7.24.3.14 off_queued 63
7.24.3.15 set_recording 63
7.24.3.16 set_snap_tick 63
7.24.3.17 set_quantized_rec 63
7.24.3.18 set_thru 63
7.24.3.19 is_dirty_main 63
7.24.3.20 is_dirty_edit 63
7.24.3.21 is_dirty_perf 63
7.24.3.22 is_dirty_names 63
7.24.3.23 set_dirty_mp 64
7.24.3.24 set_dirty 64
7.24.3.25 set_midi_channel 64
7.24.3.26 print 64
7.24.3.27 print_triggers 64
7.24.3.28 play 64
7.24.3.29 set_orig_tick 64
7.24.3.30 add_event 64
7.24.3.31 add_trigger 64
7.24.3.32 split_trigger 65
7.24.3.33 grow_trigger 65

7.24.3.34 del_trigger 65
7.24.3.35 intersectTriggers 65
7.24.3.36 intersectNotes 65
7.24.3.37 intersectEvents 66
7.24.3.38 move_selected_triggers_to 66
7.24.3.39 get_selected_trigger_start_tick 66
7.24.3.40 get_selected_trigger_end_tick 66
7.24.3.41 get_max_trigger 66
7.24.3.42 move_triggers 67
7.24.3.43 copy_triggers 67
7.24.3.44 clear_triggers 67
7.24.3.45 set_midi_bus 67
7.24.3.46 set_master_midi_bus 67
7.24.3.47 select_note_events 67
7.24.3.48 select_events 68
7.24.3.49 select_events 68
7.24.3.50 get_num_selected_notes 68
7.24.3.51 get_num_selected_events 68
7.24.3.52 select_all 68
7.24.3.53 copy_selected 68
7.24.3.54 paste_selected 68
7.24.3.55 add_note 68
7.24.3.56 add_event 69
7.24.3.57 stream_event 69
7.24.3.58 change_event_data_range 69
7.24.3.59 increment_selected 69
7.24.3.60 decrement_selected 70
7.24.3.61 grow_selected 70
7.24.3.62 stretch_selected 70
7.24.3.63 remove_marked 70
7.24.3.64 mark_selected 70
7.24.3.65 unpaint_all 70
7.24.3.66 unselect 70
7.24.3.67 verify_and_link 70
7.24.3.68 link_new 70
7.24.3.69 zero_markers 71
7.24.3.70 play_note_on 71
7.24.3.71 play_note_off 71
7.24.3.72 off_playing_notes 71
7.24.3.73 reset_draw_marker 71

7.24.3.74 get_next_note_event 71
7.24.3.75 get_next_event 71
7.24.3.76 get_next_event 71
7.24.3.77 fill_container 71
7.24.3.78 transpose_notes 72
7.25 seq64::trigger Class Reference 72
7.25.1 Detailed Description 72
7.26 user_instrument Class Reference 72
7.26.1 Detailed Description 73
7.26.2 Member Function Documentation 73
7.26.2.1 set_defaults 73
7.26.2.2 set_global 73
7.26.2.3 get_global 73
7.26.2.4 controller_max 73
7.26.2.5 controller_name 73
7.26.2.6 controller_active 74
7.26.2.7 set_controller 74
7.27 user_instrument_t Struct Reference 74
7.28 user_midi_bus Class Reference 74
7.28.1 Detailed Description 75
7.28.2 Member Function Documentation 75
7.28.2.1 set_defaults 75
7.28.2.2 set_global 75
7.28.2.3 get_global 75
7.28.2.4 channel_count 75
7.28.2.5 channel_max 76
7.28.2.6 instrument 76
7.28.2.7 set_instrument 76
7.29 user_midi_bus_t Struct Reference 76
7.30 user_settings Class Reference 76
7.30.1 Detailed Description 79
7.30.2 Member Function Documentation 79
7.30.2.1 set_defaults 79
7.30.2.2 set_globals 79
7.30.2.3 get_globals 79
7.30.2.4 bus 79
7.30.2.5 instrument 79
7.30.2.6 bus_instrument 79
7.30.2.7 mainwnd_rows 79
7.30.2.8 mainwnd_cols 79

7.30.2.9 max_sets 79
7.30.2.10 text_x 80
7.30.2.11 text_y 80
7.30.2.12 mainwid_border 80
7.30.2.13 mainwid_spacing 80
7.30.2.14 control_height 80
7.30.2.15 dump_summary 80
7.31 seq64::userfile Class Reference 80
7.31.1 Member Function Documentation 81
7.31.1.1 parse 81
7.31.1.2 write 82

Index **83**

Chapter 1

Sequencer64

Author(s) Chris Ahlstrom 2015-09-10

1.1 Introduction

Sequencer64 is a minor cleanup, refactoring, and documentation of the Seq24 live-play MIDI sequencer.

The current document describes the functions, classes, modules, and other entities used in this project.

For now, please read the ROADMAP and README files to understand the genesis of this project.

Also, we have pretty deeply documented *Seq24* and *Sequencer64* with PDF files that can be generated by git-cloning the following projects, installing a number of tools related to PDF and LaTeX, and running "make":

- `https://github.com/ahlstromcj/sequencer24-doc.git`

In the present document, we've left out a fair amount of side-material to cut down on the size of the document. For example, the main module, redundant Windows support, utility headers like easy_macros.h, simple stuff like the mutex module, the fruity variants (at least the ones already refactored into their own modules), etc., are all left out.

Chapter 2

User Testing of Sequencer64 with Yoshimi

Author(s) Chris Ahlstrom 2015-10-11

2.1 Introduction

This section describes user testing of Sequencer64 using Yoshimi. It will expand as we work our way through all the many use-cases that can be achieved with Sequencer64 and Yoshimi.

2.2 Smoke Test

Every so often we run Sequencer64 with a software synthesizer to make sure we haven't broken any functionality via our major refactoring efforts. We call it a "smoke test". We fire up the two application, and see if anything smokes.

This smoke test sets up Yoshimi with a very simple ALSA setup, and no instruments are loaded. Instead, only the "Simple Sound" is used on all channels. We've been doing this test with Yoshimi 1.3.6. The current Debian Sid ("testing") version of Yoshimi is 1.3.6-2, pulled from SourceForge. It seems to have issues, so we've been cloning and pulling the code from:

```
https://github.com/Yoshimi/yoshimi.git
```

After getting the application build and installed, the next step is to run it, using ALSA for MIDI and for audio:

```
$ yoshimi -a -A &
```

Next, fix up the configuration files for Sequencer64, `~/.config/sequencer64/sequencer64.rc` and `~/.config/sequencer64/sequencer64.usr`.

First hide `sequencer64.usr` somewhere, or delete it, as it will determine what MIDI devices are available, and we don't want that (yet). Second, make sure that `sequencer64.rc` makes the following setting:

```
[manual-alsa-ports]

# Set to 1 if you want seq24 to create its own ALSA ports and
# not connect to other clients

0   # number of manual ALSA ports
```

Next, run the newly-built version of Sequencer64:

```
$ sequencer64/sequencer64 &
```

In *File / Options / MIDI Clock*, observe the MIDI inputs made available by your system. Our system shows:

```
[0] 14:0 (Midi Through Port-0)
[1] 128:0 (TiMidity port 0)
[2] 128:0 (TiMidity port 1)
[3] 128:0 (TiMidity port 2)
[4] 128:0 (TiMidity port 3)
[5] 129:0 (input)
```

For some reason (a bug?), input "[5]" doesn't indicate that it is Yoshimi, but it is. Take note of that input number... that is the MIDI buss number that is needed to drive Yoshimi.

Also make sure that of the clock settings for those busses are "Off".

Now open the file `sequencer64/contrib/midi/b4uacuse-GM-format.midi` in Sequencer64. For all of the patterns (slots) that have lots of data in them, right click on the pattern and select *Midi Bus / [5] 129:0 (input)* and the desired channel number. (Doesn't matter much, just use up the lower channel numbers first).

Back in Yoshimi, select each Part corresponding to the channels you selected. Make sure *Enabled* is checked for each desired channel.

Back in Sequencer64, click on each pattern you want to hear, which highlights them in black. Now click the play button (green triangle). The song should play, with each part using the "Simple Sound". Not too bad for a bunch of sine waves, eh?

Now we can test the application more fully. Note that the instructions here are very light. Detailed instructions on the usage of Sequencer64 can be found in the following project, which contains a PDF file and the LaTeX code used to build it:

```
https://github.com/ahlstromcj/sequencer24-doc.git
```

Although it applies to an earlier version of the project, it still mostly holds true for Sequencer64.

2.3 Tests in the Patterns Window

Empty tracks (i.e. title-only tracks) are highlighted in yellow.

2.3.1 Patterns Window Key Shortcuts

2.3.2 The Sequencer64 User File

Chapter 3

Licenses

Library This application and its libraries, sub-applications, and documents.

Author(s) Chris Ahlstrom 2015-09-10

3.1 License Terms for the This Project.

Wherever the tag $XPC_SUITE_GPL_LICENSE$ appears, or wherever reference to the GPL licensing scheme (any version) is mentioned, substitute the appropriate license text, depending on whether the project is a library, application, documentation, or server software. We're not going to include paragraphs of licensing information in every module; you are responsible for coming here to read the licensing information.

These licenses apply to each sub-project and file artifact in the project with which this license description was packaged.

Wherever the term **XPC** is encountered in this project, it refers to my projects, which go beyond the package that contains this document.

3.2 XPC Application License

The **XPC** application license is either the **GNU GPLv2**. or the **GNU GPLv3**. Generally, projects that originate with me use the latter language, while projects I have extended may specify the former license.

3.3 XPC Library License

The **XPC** library license is the **GNU LGPLv3**.

Copyright (C) 2015-2015 by Chris Ahlstrom

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Lesser Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU LGPL version 3 license can also be found here:

http://www.gnu.org/licenses/lgpl-3.0.txt

3.4 XPC Documentation License

The **XPC** documentation license is the **GNU FDLv1.3**.

Copyright (C) 2015-2015 by Chris Ahlstrom

This documentation is free documentation; you can redistribute it and/or modify it under the terms of the GNU Free Documentation License as published by the Free Software Foundation; either version 1.3 of the License, or (at your option) any later version.

This documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Free Documentation License for more details.

You should have received a copy of the GNU Free Documentation License along with this documentation; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU FDL version 1.3 license can also be found here:

http://www.gnu.org/licenses/fdl.txt

3.5 XPC Affero License

The **XPC** "Affero" license is the **GNU AGPLv3**.

Copyright (C) 2015-2015 by Chris Ahlstrom

This server software is free server software; you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation; either version 1.3 of the License, or (at your option) any later version.

This documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Free Documentation License for more details.

You should have received a copy of the GNU Affero General Public License along with this server software; if not, write to the

```
   Free Software Foundation, Inc.
   51 Franklin Street, Fifth Floor
   Boston, MA 02110-1301, USA.
```

The text of the GNU AGPL version 3 license can also be found here:

http://www.gnu.org/licenses/agpl-3.0.txt

At the present time, no **XPC** project uses the "Affero" license.

3.6 XPC License Summary

Include one of these licenses in your Doxygen documentation with one of the following Doxygen tags specified above:

```
      \ref gpl_license_subproject
      \ref gpl_license_application
      \ref gpl_license_library
      \ref gpl_license_documentation
      \ref gpl_license_affero
```

For more information on navigating GNU licensing, see this page:

```
  http://www.gnu.org/licenses/
```

Copies of these licenses (and some logos) are provided in the `licenses` directory of the main project (or you can search for them at *gnu.org*).

Chapter 4

Todo List

File globals.h

There are additional user-interface and MIDI scaling variables in the perfroll module that we need to move here.

Global seq64::perform::set_bpm (int a_bpm)

I think this logic is wrong, in that it needs only one of the two to be stopped before it sets the BPM, while it seems to me that both should be stopped; to be determined.

Global seq64::perform::start_playing (bool flag=false)

Verify the usage and nature of this flag.

Global user_settings::bus_instrument (int buss, int channel)

Do this for controllers values and for user_instrument members.

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

seq64::automutex 15
seq64::click 15
seq64::configfile 18
  seq64::optionsfile 40
  seq64::userfile 80
seq64::event 19
seq64::event_list 23
seq64::gui_assistant 25
seq64::gui_play_base 26
seq64::jack_assistant 26
seq64::jack_scratchpad 28
seq64::keys_perform 28
seq64::keys_perform_transfer 31
seq64::keystroke 31
seq64::lash 32
seq64::midi_container 33
  seq64::midi_list 35
  seq64::midi_vector 37
seq64::midifile 38
seq64::mutex 39
  seq64::condition_var 17
seq64::perform 42
seq64::performcallback 53
rc_settings 53
seq64::sequence 55
seq64::trigger 72
user_instrument 72
user_instrument_t 74
user_midi_bus 74
user_midi_bus_t 76
user_settings 76

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

seq64::automutex
Provides a mutex that locks automatically when created, and unlocks when destroyed . . . . . ??
seq64::click
Encapsulates any possible mouse click . . . . ??
seq64::condition_var
A mutex works best in conjunction with a condition variable . . . . ??
seq64::configfile
This class is the abstract base class for optionsfile and userfile . . . . ??
seq64::event
Provides events for management of MIDI events . . . . ??
seq64::event_list
Receptable for MIDI events . . . . ??
seq64::gui_assistant
This class provides an interface for some of the GUI support needed in Sequencer64 . . . . . ??
seq64::gui_play_base
This class provides an interface for basic GUI support . . . . ??
seq64::jack_assistant
This class provides the performance mode JACK support . . . . ??
seq64::jack_scratchpad
Provide a temporary structure for passing data and results between a perform and jack_assistant object . . . . ??
seq64::keys_perform
This class supports the performance mode . . . . ??
seq64::keys_perform_transfer
Provides a data-transfer structure to make it easier to fill in a keys_perform object's members using sscanf() . . . . ??
seq64::keystroke
Encapsulates any practical keystroke . . . . ??
seq64::lash
This class supports LASH operations, if compiled with LASH support (i.e . . . . ??
seq64::midi_container
This class is the abstract base class for a container of MIDI track information . . . . ??
seq64::midi_list
This class is the std::list implementation of the midi_container . . . . ??
seq64::midi_vector
This class is the std::vector implementation of the midi_container . . . . ??
seq64::midifile
This class handles the parsing and writing of MIDI files . . . . ??

seq64::mutex
Simple wrapper for the pthread_mutex_t type used as a recursive mutex . . . . ??
seq64::optionsfile
Provides a file for reading and writing the application' main configuration file . . . . ??
seq64::perform
This class supports the performance mode . . . . ??
seq64::performcallback
Provides for notification of events . . . . ??
rc_settings
This class contains the options formerly named "global_xxxxxx" . . . . ??
seq64::sequence
Firstly a receptable for a single track of MIDI data read from a MIDI file or edited into a pattern ??
seq64::trigger
This class is used in playback . . . . ??
user_instrument
Provides data about the MIDI instruments, readable from the "user" configuration file . . . . . ??
user_instrument_t
This structure corresponds to `[user-instrument-N]` definitions in the `~/.seq24usr` or `~/.config/sequencer64/sequencer64.rc` file . . . . ??
user_midi_bus
Provides data about the MIDI busses, readable from the "user" configuration file . . . . ??
user_midi_bus_t
This structure corresponds to `[user-midi-bus-0]` definitions in the `~/.seq24usr` ("user") file . . . . ??
user_settings
Holds the current values of sequence settings and settings that can modify the number of sequences and the configuration of the user-interface . . . . ??
seq64::userfile
Supports the user's ~/.seq24usr configuration file . . . . ??

Chapter 7

Data Structure Documentation

7.1 seq64::automutex Class Reference

Provides a mutex that locks automatically when created, and unlocks when destroyed.

Public Member Functions

- automutex (mutex &my_mutex)

    Principal constructor gets a reference to a mutex parameter, and then locks the mutex.
- ∼automutex ()

    The destructor unlocks the mutex.

7.1.1 Detailed Description

This has a couple of benefits. First, it is more threadsafe in the face of exception handling. Secondly, it can be done with just one line of code.

7.2 seq64::click Class Reference

Encapsulates any possible mouse click.

Public Member Functions

- click ()

    The constructor for class click.
- click (int x, int y, int button=SEQ64_CLICK_BUTTON_LEFT, bool press=true, seq_modifier_t modkey=SE↩
Q64_NO_MASK)

    Principal constructor for class click.
- click (const click &rhs)

    Provides a stock copy constructor.
- click & operator= (const click &rhs)

    Provides a stock principal assignment operator.
- bool is_press () const

    'Getter' function for member m_is_press
- bool is_left () const

    'Getter' function for member m_button to test for left, right, and middle buttons.

- int x () const
  'Getter' function for member m_x
- int y () const
  'Getter' function for member m_y
- int button () const
  'Getter' function for member m_button
- seq_modifier_t modifier () const
  'Getter' function for member m_modifier
- bool mod_control () const
  'Getter' function for member m_modifier tested for Ctrl key.
- bool mod_control_shift () const
  'Getter' function for member m_modifier tested for Ctrl and Shift key.
- bool mod_super () const
  'Getter' function for member m_modifier tested for Mod4/Super/Windows key.

7.2.1 Detailed Description

Useful in passing more generic events to non-GUI classes.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 seq64::click::click ( )

Sets all members to false, zero, or the lowest good value.

7.2.2.2 seq64::click::click (int *x*, int *y*, int *button* = `SEQ64_CLICK_BUTTON_LEFT`, bool *press* = `true`, seq_modifier_t *modkey* = `SEQ64_NO_MASK`)

This function is the only way to set value for the click members (other than the copy constructor and principal assignment operator.

Parameters

x	The putative x value of the button click.
y	The putative y value of the button click.
button	The value of the button that was clicked, set to 1, 2, or 3.
press	Set to true if the event was a button press, false if it was a button release.
modkey	Indicates which modifier key (such as Ctrl or Alt), if any, was pressed at the same time as the click action.

7.2.2.3 seq64::click::click (const click & *rhs*)

It is nice to be explicit about these kinds of functions, even if it gets tedious.

Parameters

rhs	Provies the source object to be copied.

7.2.3 Member Function Documentation

7.2.3.1 click & seq64::click::operator= (const click & *rhs*)

It is nice to be explicit about these kinds of functions, even if it gets tedious.

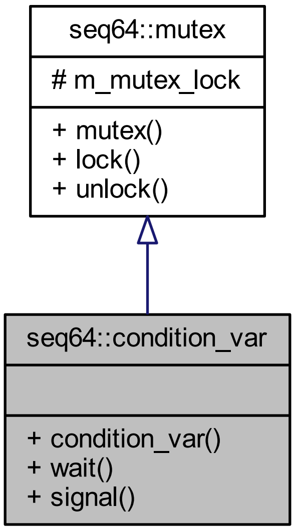
Parameters

rhs	Provies the source object to be assigned from. The assignment is not made if "this" has the same address as this parameter.

7.3 seq64::condition_var Class Reference

A mutex works best in conjunction with a condition variable.

Inheritance diagram for seq64::condition_var:



Public Member Functions

- condition_var ()

  Initialize the condition variable with the global variable.
- void wait ()

  Waits for the confition variable.
- void signal ()

  Signals the confition variable.

Additional Inherited Members

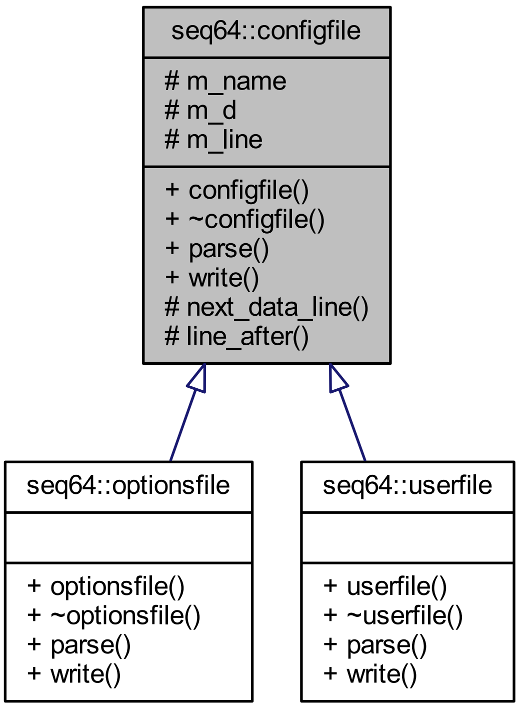
7.3.1 Detailed Description

Therefore this class derives from the mutex class. A "has-a" relationship might be more logical than this "is-a" relationship.

7.4 seq64::configfile Class Reference

This class is the abstract base class for optionsfile and userfile.

Inheritance diagram for seq64::configfile:



Public Member Functions

- configfile (const std::string &a_name)

  Provides the string constructor for a configuration file.
- virtual ∼configfile ()

  A rote constructor needed for a base class.

Protected Member Functions

- void next_data_line (std::ifstream &a_file)

  Gets the next line of data from an input stream.
- void line_after (std::ifstream &a_file, const std::string &a_tag)

  This function gets a specific line of text, specified as a tag.

Protected Attributes

- std::string m_name

  Provides the name of the file.

- unsigned char ∗ m_d

  Points to an allocated buffer that holds the data for the configuration file.
- char m_line [SEQ64_LINE_MAX]

  The current line of text being processed.

7.4.1 Constructor & Destructor Documentation

7.4.1.1 seq64::configfile::configfile (const std::string & *name*)

Parameters

name	The name of the configuration file.

7.4.2 Member Function Documentation

7.4.2.1 void seq64::configfile::next_data_line (std::ifstream & *file*) [protected]

If the line starts with a number-sign, a space (!), or a null, it is skipped, to try the next line. This occurs until an EOF is encountered.

We may try to convert this item to a reference; pointers can be subject to problems. For example, what if someone passes a nullpointer? For speed, we don't check it.

Member m_line is a "global" return value.

Parameters

a_file	Points to an input stream.

7.4.2.2 void seq64::configfile::line_after (std::ifstream & *file,* const std::string & *tag*) [protected]

Parameters

file	Points to the input file stream.
tag	Provides a tag to be found. Lines are read until a match occurs with this tag.

7.4.3 Field Documentation

7.4.3.1 char seq64::configfile::m_line[SEQ64_LINE_MAX] [protected]

This member receives an input line, and so needs to be a character buffer.

7.5 seq64::event Class Reference

Provides events for management of MIDI events.

Public Member Functions

- event ()

  This constructor simply initializes all of the class members.
- ∼event ()

  This destructor explicitly deletes m_sysex and sets it to null.

- bool operator< (const event &rhsevent) const

  If the current timestamp equal the event's timestamp, then this function returns true if the current rank is less than the event's rank.
- void set_timestamp (unsigned long time)

  'Setter' function for member m_timestamp
- long get_timestamp () const

  'Getter' function for member m_timestamp
- unsigned char status () const

  'Getter' function for member m_status
- void mod_timestamp (unsigned long a_mod)

  Calculates the value of the current timestamp modulo the given parameter.
- void set_status (char status)

  Sets the m_status member to the value of a_status.
- unsigned char get_status () const

  'Getter' function for member m_status
- void set_data (char d1)

  Clears the most-significant-bit of the d1 parameter, and sets it into the first byte of m_data.
- void set_data (char d1, char d2)

  Clears the most-significant-bit of both parameters, and sets them into the first and second bytes of m_data.
- void get_data (unsigned char &d0, unsigned char &d1)

  Retrieves the two data bytes from m_data[] and copies each into its respective parameter.
- void increment_data1 ()

  Increments the first data byte (m_data[1]) and clears the most significant bit.
- void decrement_data1 ()

  Decrements the first data byte (m_data[1]) and clears the most significant bit.
- void increment_data2 ()

  Increments the second data byte (m_data[1]) and clears the most significant bit.
- void decrement_data2 ()

  Decrements the second data byte (m_data[1]) and clears the most significant bit.
- void start_sysex ()

  Deletes and clears out the SYSEX buffer.
- bool append_sysex (unsigned char ∗data, long size)

  Appends SYSEX data to a new buffer.
- unsigned char ∗ get_sysex () const

  'Getter' function for member m_sysex
- void set_size (long a_size)

  'Setter' function for member m_size
- long get_size () const

  'Getter' function for member m_size
- void link (event ∗a_event)

  Sets m_has_link and sets m_link to the provided event pointer.
- event ∗ get_linked () const

  'Getter' function for member m_linked
- bool is_linked () const

  'Getter' function for member m_has_link
- void clear_link ()

  'Setter' function for member m_has_link
- void paint ()

  'Setter' function for member m_painted
- void unpaint ()

  'Setter' function for member m_painted

- bool is_painted () const

  'Getter' function for member m_painted
- void mark ()

  'Setter' function for member m_marked
- void unmark ()

  'Setter' function for member m_marked
- bool is_marked () const

  'Getter' function for member m_marked
- void select ()

  'Setter' function for member m_selected
- void unselect ()

  'Setter' function for member m_selected
- bool is_selected () const

  'Getter' function for member m_selected
- void make_clock ()

  Sets m_status to EVENT_MIDI_CLOCK;.
- unsigned char data (int index) const

  'Getter' function for member m_data[]
- unsigned char get_note () const

  Assuming m_data[] holds a note, get the note number, which is in the first data byte, m_data[0].
- void set_note (char a_note)

  Sets the note number, clearing off the most-significant-bit and assigning it to the first data byte, m_data[0].
- unsigned char get_note_velocity () const

  'Getter' function for member m_data[1], the note velocity.
- void set_note_velocity (int a_vel)

  Sets the note velocity, with is held in the second data byte, m_data[1].
- bool is_note_on () const

  Returns true if m_status is EVENT_NOTE_ON.
- bool is_note_off () const

  Returns true if m_status is EVENT_NOTE_OFF.
- void print ()

  Prints out the timestamp, data size, the current status byte, any SYSEX data if present, or the two data bytes for the status byte.
- int get_rank () const

  This function is used in sorting MIDI status events (e.g.

7.5.1 Detailed Description

A MIDI event consists of 3 bytes:

```
-#  Status byte, 1sssnnn, where the sss bits specify the type of
    message, and the nnnn bits denote the channel number.
    The status byte always starts with 0.
-#  The first data byte, 0xxxxxxx, where the data byte always
    start with 0, and the xxxxxxx values range from 0 to 127.
-#  The second data byte, 0xxxxxxx.
```

This class may have too many member functions.

7.5.2 Member Function Documentation

7.5.2.1 bool seq64::event::operator< (const event & *rhs*) const

Otherwise, it returns true if the current timestamp is less than the event's timestamp.

Warning

The less-than operator is supposed to support a "strict weak ordering", and is supposed to leave equivalent values in the same order they were before the sort. However, every time we load and save our sample MIDI file, events get reversed. Here are program-changes that get reversed:

```
    Save N:     0070: 6E 00 C4 48 00 C4 0C 00  C4 57 00 C4 19 00 C4 26
    Save N+1:   0070: 6E 00 C4 26 00 C4 19 00  C4 57 00 C4 0C 00 C4 48


The 0070 is the offset within the versions of the
b4uacuse-seq24.midi file.

Because of this mis-feature, and the very slow speed of loading a
MIDI file when Sequencer64 is built for debugging, we are
exploring using an std::map instead of an std::list.  Search for
occurrences of the USE_EVENT_MAP macro. (This actually works better
than a list, we have found).
```

Parameters

rhs	The object to be compared against.

Returns

Returns true if the time-stamp and "rank" are less than those of the comparison object.

7.5.2.2 void seq64::event::mod_timestamp (unsigned long *a_mod*) [inline]

Parameters

a_mod	The value to mod the timestamp against.

Returns

Returns a value ranging from 0 to a_mod-1.

7.5.2.3 void seq64::event::set_status (char *status*)

If a_status is a non-channel event, then the channel portion of the status is cleared using a bitwise AND against EVENT_CLEAR_CHAN_MASK..

7.5.2.4 void seq64::event::set_data (char *d1*)

Parameters

d1	The byte value to set. We should make these all "midibytes".

7.5.2.5 void seq64::event::set_data (char *d1*, char *d2*)

Parameters

d1	The first byte value to set. We should make these all "midibytes".
d2	The second byte value to set. We should make these all "midibytes".

7.5.2.6 void seq64::event::get_data (unsigned char & *d0*, unsigned char & *d1*)

Parameters

d0	[out] The return reference for the first byte.
d1	[out] The return reference for the first byte.

7.5.2.7 bool seq64::event::append_sysex (unsigned char ∗ *a_data*, long *a_size*)

First, a buffer of size m_size+a_size is created. The existing SYSEX data (stored in m_sysex) is copied to this buffer. Then the data represented by a_data and a_size is appended to that data buffer. Then the original SYSEX buffer, m_sysex, is deleted, and m_sysex is assigned to the new buffer..

Warning

This function does not check any pointers.

Parameters

a_data	Provides the additional SYSEX data.
a_size	Provides the size of the additional SYSEX data.

Returns

Returns false if there was an EVENT_SYSEX_END byte in the appended data.

7.5.2.8 int seq64::event::get_rank ( ) const

The ranking, from high to low, is note off, note on, aftertouch, channel pressure, and pitch wheel, control change, and program changes.

note on/off, aftertouch, control change, etc.) The sort order is not determined by the actual status values.

The lower the ranking the more upfront an item comes in the sort order.

Returns

Returns the rank of the current m_status byte.

7.6 seq64::event_list Class Reference

The event_list class is a receptable for MIDI events.

Public Member Functions

- event_list ()

  Principal constructor.
- event_list (const event_list &a_rhs)

Copy constructor.

- event_list & operator= (const event_list &a_rhs)

  Principal assignment operator.
- ∼event_list ()

  A rote destructor.
- iterator begin ()

  'Getter' function for member m_events.begin(), non-constant version.
- const_iterator begin () const

  'Getter' function for member m_events.begin(), constant version.
- iterator end ()

  'Getter' function for member m_events.end(), non-constant version.
- const_iterator end () const

  'Getter' function for member m_events.end(), constant version.
- int count () const

  Returns the number of events stored in m_events.
- void add (const event &e, bool postsort=true)

  Adds an event to the internal event list in an optionally sorted manner.
- void remove (iterator ie)

  Provides a wrapper for the iterator form of erase(), which is the only one that sequence uses.
- void clear ()

  Provides a wrapper for clear().
- void sort ()

  Wrapper for std::list::sort(), or, since multimaps are always sorted, an empty function.

Static Public Member Functions

- static event & dref (iterator ie)

  Dereference access for list or map.
- static const event & dref (const_iterator ie)

  Dereference const access for list or map.

7.6.1 Detailed Description

Two implementations, an std::multimap, and the original, an std::list, are provided for comparison, and are selected at build time, by manually defining the USE_EVENT_MAP macro near the top of this module.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 seq64::event_list::event_list (const **event_list** & *rhs*)

Parameters

rhs	Provides the event list to be copied.

7.6.3 Member Function Documentation

7.6.3.1 **event_list** & seq64::event_list::operator= (const **event_list** & *rhs*)

Follows the stock rules for such an operator, just assigning member values.

Parameters

rhs	Provides the event list to be assigned.

7.6.3.2 int seq64::event_list::count ( ) const [inline]

We like returning an integer instead of size_t, and rename the function so nobody is fooled.

7.6.3.3 void seq64::event_list::add (const event & *e*, bool *postsort* = true)

It is a wrapper, wrapper for insert() or push_front(), with an option to call sort().

For the std::multimap implementation, This is an option if we want to make sure the insertion succeed.

```
std::pair<Events::iterator, bool> result = m_events.insert(p);
return result.second;
```

Warning

This pushing (and, in writing the MIDI file, the popping), causes events with identical timestamps to be written in reverse order. Doesn't affect functionality, but it's puzzling until one understands what is happening. That's why we're exploring using a multimap as the container.

Parameters

e	Provides the event to be added to the list.
postsort	If true, and the std::list implementation has been built in, then the event list is sorted after the addition. This is a time-consuming operation.

7.7 seq64::gui_assistant Class Reference

This class provides an interface for some of the GUI support needed in Sequencer64.

Public Member Functions

- gui_assistant (keys_perform &kp)

  This constructor wires in some externally (for now) created objects.
- const keys_perform & keys () const

  'Getter' function for member m_keys_perform The const getter.
- keys_perform & keys ()

  'Getter' function for member m_keys_perform The un-const getter.

7.7.1 Detailed Description

It also contain a number of helper objects that all kind of go together; only this assistant object will need to be passed around (by non-GUI code).

7.7.2 Constructor & Destructor Documentation

7.7.2.1 seq64::gui_assistant::gui_assistant (keys_perform & *kp*)

Parameters

kp	Provides a set of key codes to be used by the perform object to control patterns and their performance.

7.8 seq64::gui_play_base Class Reference

This class provides an interface for basic GUI support.

Protected Member Functions

- virtual bool do_realize_event ()

    Do-nothing interface function that might not need to be overridden in many classes.
- virtual bool do_expose_event ()

    Do-nothing interface function that might not need to be overridden in many classes.
- virtual bool do_focus_in_event ()

    Do-nothing interface function that might not need to be overridden in many classes.
- virtual bool do_focus_out_event ()

    Do-nothing interface function that might not need to be overridden in many classes.
- virtual bool do_motion_notify_event (click &)

    Do-nothing interface function that might not need to be overridden in many classes.
- virtual bool do_delete_event ()

    Do-nothing interface function that might not need to be overridden in many classes.

7.8.1 Detailed Description

Much is to be determined at this point.

7.9 seq64::jack_assistant Class Reference

This class provides the performance mode JACK support.

Public Member Functions

- jack_assistant (perform &parent)

    This constructor initializes a number of member variables, some of them public!
- ∼jack_assistant ()

    The destructor doesn't need to do anything yet.
- bool is_running () const

    'Getter' function for member m_jack_running
- bool is_master () const

    'Getter' function for member m_jack_master
- perform & parent ()

    'Getter' function for member m_jack_parent Needed for external callbacks.
- bool init ()

    Initializes JACK support.
- void deinit ()

    Tears down the JACK infrastructure.

- void start ()

  If JACK is supported, starts the JACK transport.
- void stop ()

  If JACK is supported, stops the JACK transport.
- void position (bool a_state)

  If JACK is supported and running, sets the position of the transport.
- bool output (jack_scratchpad &pad)

  Performance output function for JACK, called by the perform function of the same name.

Friends

- int jack_sync_callback (jack_transport_state_t state, jack_position_t ∗pos, void ∗arg)

  Global functions for JACK support and JACK sessions.
- void jack_shutdown (void ∗arg)

  Shutdown JACK by clearing the perform::m_jack_running flag.
- void jack_timebase_callback (jack_transport_state_t state, jack_nframes_t nframes, jack_position_t ∗pos, int new_pos, void ∗arg)

  This function sets the JACK position structure.

7.9.1 Constructor & Destructor Documentation

7.9.1.1 seq64::jack_assistant::jack_assistant (perform & *parent*)

Parameters

parent	Provides a reference to the main perform object that needs to control JACK event.

7.9.2 Member Function Documentation

7.9.2.1 bool seq64::jack_assistant::init ( )

Then we become a new client of the JACK server.

Who calls this routine?

Returns

Returns true if JACK is now considered to be running (or if it was already running.)

7.9.2.2 void seq64::jack_assistant::position (bool *a_state*)

Warning

A lot of this code is effectively disabled by an early return statement.

Parameters

state	If true, the current tick is set to the leftmost tick.

7.9.2.3 bool seq64::jack_assistant::output (jack_scratchpad & *pad*)

Parameters

pad	Provide a JACK scratchpad, whatever that is.

Returns

Returns true if JACK is running.

7.9.3 Friends And Related Function Documentation

7.9.3.1 int jack_sync_callback (jack_transport_state_t *state*, jack_position_t ∗ *pos*, void ∗ *arg*) `[friend]`

This JACK synchronization callback informs the specified perform object of the current state and parameters of JACK.

Parameters

state	The JACK Transport state.
pos	The JACK position value.
arg	The pointer to the perform object. Currently not checked for nullity.

7.9.3.2 void jack_shutdown (void ∗ *arg*) `[friend]`

Parameters

arg	Points to the jack_assistant in charge of JACK support for the perform object.

7.9.3.3 void jack_timebase_callback (jack_transport_state_t *state*, jack_nframes_t *nframes*, jack_position_t ∗ *pos*, int *new_pos*, void ∗ *arg*) `[friend]`

Parameters

state	Indicates the current state of JACK transport.
nframes	The number of JACK frames.
pos	Provides the position structure to be filled in.
new_pos	The new positions to be set.
arg	Provides the jack_assistant pointer, currently unchecked for nullity.

7.10 seq64::jack_scratchpad Struct Reference

Provide a temporary structure for passing data and results between a perform and jack_assistant object.

7.10.1 Detailed Description

The jack_assistant class already has access to the members of perform, but it needs access to and modification of local variables in perform::output_func().

7.11 seq64::keys_perform Class Reference

This class supports the performance mode.

Public Member Functions

- keys_perform ()

    This construction initializes a vast number of member variables, some of them public!
- ∼keys_perform ()

    The destructor sets some running flags to false, signals this condition, then joins the input and output threads if the were launched.
- void set_keys (const keys_perform_transfer &kpt)

    Copies fields from the transfer structure in this object.
- void get_keys (keys_perform_transfer &kpt)

    Copies fields from this object into the transfer structure.
- bool show_ui_sequence_key () const

    ***Accessor** m_key_show_ui_sequency_key*
- virtual std::string key_name (unsigned int key) const

    Obtains the name of the key.
- virtual void set_all_key_events ()

    Provides base class functionality.
- virtual void set_all_key_groups ()

    Provides base class functionality.
- void set_key_event (unsigned int keycode, long sequence_slot)

    At construction time, this function sets up one keycode and one event slot.
- void set_key_group (unsigned int keycode, long group_slot)

    At construction time, this function sets up one keycode and one group slot.

Protected Types

- typedef std::map< unsigned int, long > SlotMap

    This typedef defines a map in which the key is the keycode, that is, the integer value of a keystroke, and the value is the pattern/sequence number or slot.
- typedef std::map< long, unsigned int > RevSlotMap

    This typedef is like SlotMap, but used for lookup in the other direction.

7.11.1 Detailed Description

It has way too many data members, many of the public. Might be ripe for refactoring.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 seq64::keys_perform::∼keys_perform ( )

Finally, any active patterns/sequences are deleted.

7.11.3 Member Function Documentation

7.11.3.1 void seq64::keys_perform::set_keys (const keys_perform_transfer & *kpt*)

This structure holds all of the key settings from the File / Options / Keyboard tab dialog.

Parameters

kpt	The structure that holds the values of the keys to be used for various purposes in controlling a performance live.

7.11.3.2 void seq64::keys_perform::get_keys (keys_perform_transfer & *kpt*)

Parameters

kpt	The structure that holds the values of the keys to be used for various purposes in controlling a performance live.

7.11.3.3 bool seq64::keys_perform::show_ui_sequence_key ( ) const `[inline]`

Used in mainwid, options, optionsfile, userfile, and perform.

7.11.3.4 std::string seq64::keys_perform::key_name (unsigned int *key*) const `[virtual]`

In gtkmm, this is done via the gdk_keyval_name() function. Here, in the base class, we just provide an easy-to-create string.

Parameters

key	Provides the numeric value of the keystroke.

Returns

Returns the name of the key, in the format "Key 0xkkkk".

7.11.3.5 virtual void seq64::keys_perform::set_all_key_events ( ) `[inline]`, `[virtual]`

Must be called by the derived-class's override of this function.

7.11.3.6 virtual void seq64::keys_perform::set_all_key_groups ( ) `[inline]`, `[virtual]`

Must be called by the derived-class's override of this function.

7.11.3.7 void seq64::keys_perform::set_key_event (unsigned int *keycode,* long *sequence_slot*)

It is called 32 times, corresponding the pattern/sequence slots in the Patterns window.

Parameters

keycode	The key to be assigned.
sequence_slot	The perform event slot into which the keycode will be assigned.

7.11.3.8 void seq64::keys_perform::set_key_group (unsigned int *keycode,* long *group_slot*)

It is called 32 times, corresponding the pattern/sequence slots in the Patterns window.

Parameters

keycode	The key to be assigned.
group_slot	The perform group slot into which the keycode will be assigned.

7.12 seq64::keys_perform_transfer Struct Reference

Provides a data-transfer structure to make it easier to fill in a keys_perform object's members using sscanf().

7.13 seq64::keystroke Class Reference

Encapsulates any practical keystroke.

Public Member Functions

- keystroke ()

  The default constructor for class keystroke.
- keystroke (unsigned int key, bool press=SEQ64_KEYSTROKE_PRESS, int modkey=int(SEQ64_NO_MAS↩
K))

  The principal constructor.
- keystroke (const keystroke &rhs)

  Provides the rote copy constructor.
- keystroke & operator= (const keystroke &rhs)

  Provides the rote principal assignment operator.
- bool is_press () const

  'Getter' function for member m_is_press
- bool is_letter (int ch=SEQ64_KEYSTROKE_BAD_VALUE) const

  'Getter' function for member m_key to test letters, handles ASCII only.
- bool is_delete () const

  m_key to test for a delete-causing key.
- unsigned int key () const

  'Getter' function for member m_key
- seq_modifier_t modifier () const

  'Getter' function for member m_modifier
- bool mod_control () const

  'Getter' function for member m_modifier tested for Ctrl key.
- bool mod_control_shift () const

  'Getter' function for member m_modifier tested for Ctrl and Shift key.
- bool mod_super () const

  'Getter' function for member m_modifier tested for Mod4/Super/Windows key.

7.13.1 Detailed Description

Useful in passing more generic events to non-GUI classes.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 seq64::keystroke::keystroke (unsigned int *key*, bool *press* = `SEQ64_KEYSTROKE_PRESS`, int *modkey* = `int(SEQ64_NO_MASK)`)

Parameters

key	The keystroke number of the key that was pressed or released.
press	If true, the keystroke action was a press, otherwise it was a release.
modkey	The modifier key combination that was pressed, if any, in the form of a bit-mask, as defined in the gdk_basic_keys module. Common mask values are SEQ64_SHIFT_MASK, SEQ64←_CONTROL_MASK, SEQ64_MOD1_MASK, and SEQ64_MOD4_MASK. If no modifier, this value is SEQ64_NO_MASK.

7.13.2.2 seq64::keystroke::keystroke (const keystroke & *rhs*)

Parameters

rhs	The object to be copied.

7.13.3 Member Function Documentation

7.13.3.1 keystroke & seq64::keystroke::operator= (const keystroke & *rhs*)

Parameters

rhs	The object to be assigned.

Returns

Returns the reference to the current object, for use in assignment chains.

7.13.3.2 bool seq64::keystroke::is_letter (int *ch* = `SEQ64_KEYSTROKE_BAD_VALUE`) const

Parameters

ch	An optional character to test as an ASCII letter.

Returns

If a character is not provided, true is returned if it is an upper or lower-case letter. Otherwise, true is returned if the m_key value matches the character case-insensitively.

Tricky Code

7.14 seq64::lash Class Reference

This class supports LASH operations, if compiled with LASH support (i.e.

Public Member Functions

- lash (perform &p, int argc, char ∗∗argv)

  This constructor calls lash_extract(), using the command-line arguments, if SEQ64_LASH_SUPPORT is enabled.
- void set_alsa_client_id (int id)

  Make ourselves a LASH ALSA client.
- void start ()

  Process any LASH events every 250 msec, which is an arbitrarily chosen interval.

7.14.1 Detailed Description

SEQ64_LASH_SUPPORT is defined). All of the #ifdef skeleton work is done in this class in such a way that any other part of the code can use this class whether or not lash support is actually built in; the functions will just do nothing.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 seq64::lash::lash (perform & *p*, int *argc*, char ** *argv*)

We fixed the crazy usage of argc and argv here and in the client code in the seq24 module.

Parameters

p	The perform object that needs to implement LASH support.
argc	The number of command-line arguments.
argv	The command-line arguments.

7.14.3 Member Function Documentation

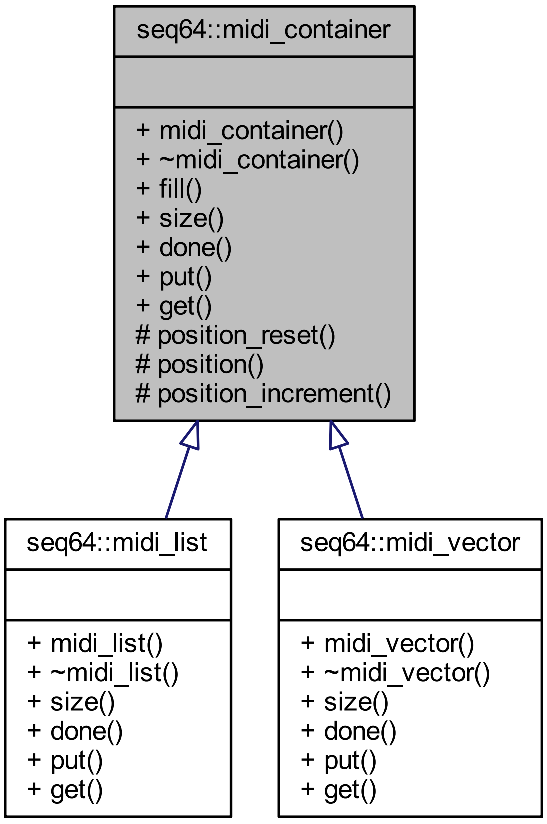
7.14.3.1 void seq64::lash::set_alsa_client_id (int *id*)

/param id The ALSA client ID to be set.

7.15 seq64::midi_container Class Reference

This class is the abstract base class for a container of MIDI track information.

Inheritance diagram for seq64::midi_container:



Public Member Functions

- midi_container (sequence &seq)

  Fills in the few members of this class.
- virtual ∼midi_container ()

  A rote constructor needed for a base class.
- void fill (int tracknumber)

  This function fills the given character list with MIDI data from the current sequence, preparatory to writing it to a file.
- virtual std::size_t size () const

  Returns the size of the container, in midibytes.
- virtual bool done () const

  Instead of checking for the size of the container when "emptying" it [see the midifile::write() function], use this function, which is overridden to match the type of container being used.
- virtual void put (midibyte b)=0

  Provides a way to add a MIDI byte into the container.
- virtual midibyte get ()=0

  Provide a way to get the next byte from the container.

Protected Member Functions

- unsigned int position () const

  Returns the current position.

7.15.1 Member Function Documentation

7.15.1.1 void seq64::midi_container::fill (int *tracknumber*)

Note that some of the events might not come out in the same order they were stored in (we see that with program-change events.

This function replaces sequence::fill_container().

Now, for sequence 0, an alternate format for writing the sequencer number chunk is "FF 00 00". But that format can only occur in the first track, and the rest of the tracks then don't need a sequence number, since it is assume to increment. This application doesn't bother with that shortcut.

Not threadsafe The sequence object bound to this container needs to provide the locking mechanism when calling this function.

Parameters

tracknumber	Provides the track number. This number is masked into the track information.

7.15.1.2 virtual void seq64::midi_container::put (midibyte *b*) `[pure virtual]`

The original seq24 container used an std::list and a push_front operation.

Implemented in seq64::midi_list, and seq64::midi_vector.

7.15.1.3 virtual midibyte seq64::midi_container::get ( ) `[pure virtual]`

It also increments m_position_for_get.

Implemented in seq64::midi_list, and seq64::midi_vector.

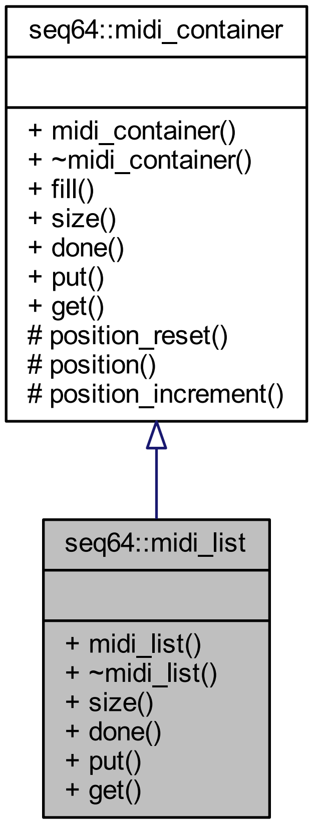
7.15.1.4 unsigned int seq64::midi_container::position ( ) const `[inline]`, `[protected]`

Before the return, the position counter is incremented to the next position.

7.16 seq64::midi_list Class Reference

This class is the std::list implementation of the midi_container.

Inheritance diagram for seq64::midi_list:



Public Member Functions

- midi_list (sequence &seq)

  This constructor fills in the members.
- virtual ∼midi_list ()

  A rote constructor needed for a base class.
- virtual std::size_t size () const

  Returns the size of the container, in midibytes.
- virtual bool done () const

  For popping data from the MIDI list, we are done when the container is empty.
- virtual void put (midibyte b)

  Provides a way to add a MIDI byte into the list.
- virtual midibyte get ()

  Provide a way to get the next byte from the container.

Additional Inherited Members

7.16.1 Member Function Documentation

7.16.1.1 virtual void seq64::midi_list::put (midibyte *b*) [inline], [virtual]

The original seq24 list used an std::list and a push_front operation.

Implements seq64::midi_container.

7.16.1.2 virtual midibyte seq64::midi_list::get ( ) [inline], [virtual]

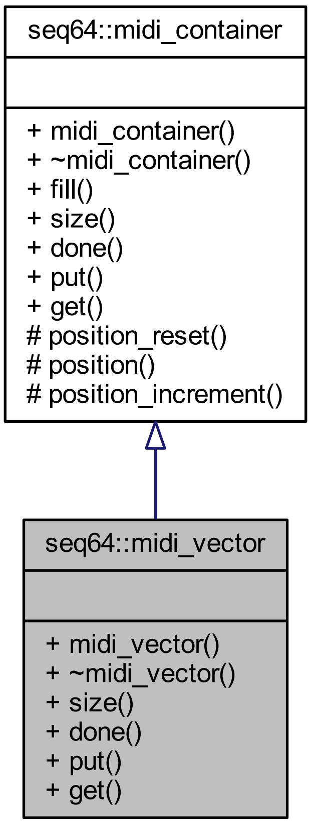
In this implement, m_position_for_get is not used. The elements of the container are popped of backward!

Implements seq64::midi_container.

7.17 seq64::midi_vector Class Reference

This class is the std::vector implementation of the midi_container.

Inheritance diagram for seq64::midi_vector:



Public Member Functions

- midi_vector (sequence &seq)

  This constructor fills in the members.

- virtual ∼midi_vector ()
  A rote constructor needed for a base class.
- virtual std::size_t size () const
  Returns the size of the container, in midibytes.
- virtual bool done () const
  For iterating through the data in the MIDI vector, we are done when we've gotten the last element of the container.
- virtual void put (midibyte b)
  Provides a way to add a MIDI byte into the list.
- virtual midibyte get ()
  Provide a way to get the next byte from the container.

Additional Inherited Members

7.17.1 Member Function Documentation

7.17.1.1 virtual void seq64::midi_vector::put (midibyte *b*) [inline], [virtual]

The original seq24 list used an std::list and a push_front operation.

Implements seq64::midi_container.

7.17.1.2 virtual midibyte seq64::midi_vector::get ( ) [inline], [virtual]

In this implement, m_position_for_get is not used. The elements of the container are popped of backward!

Implements seq64::midi_container.

7.18 seq64::midifile Class Reference

This class handles the parsing and writing of MIDI files.

Public Member Functions

- midifile (const std::string &name, bool propformat=true)
  Principal constructor.
- ∼midifile ()
  A rote destructor.
- bool parse (perform &a_perf, int a_screen_set)
  This function opens a binary MIDI file and parses it into sequences and other application objects.
- bool write (perform &a_perf)
  Write the whole MIDI data and Seq24 information out to the file.

7.18.1 Detailed Description

In addition to the standard MIDI tracks, it also handles some "private" or "proprietary" tracks specific to Seq24. It does not, however, handle SYSEX events.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 seq64::midifile::midifile (const std::string & *a_name,* bool *propformat* = true)

Parameters

a_name	Provides the name of the MIDI file to be read or written.
propformat	If true, write out the MIDI file using the MIDI-compliant sequencer-specific prefix in from of the seq24-specific SeqSpec tags defined in the globals module. This option is true by default. Note that this option is only used in writing; reading can handle either format transparently.

7.18.3 Member Function Documentation

7.18.3.1 bool seq64::midifile::parse (perform & *a_perf,* int *a_screen_set*)

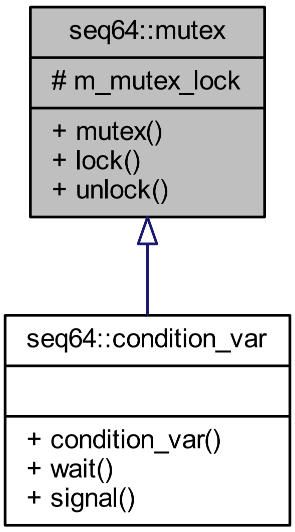
In addition to the standard MIDI track data in a normal track, Seq24 adds four sequencer-specific events just before the end of the track:

```
    c_triggers_new:     SeqSpec FF 7F 1C 24 24 00 08 00 00 ...
    c_midibus:          SeqSpec FF 7F 05 24 24 00 01 00
    c_timesig:          SeqSpec FF 7F 06 24 24 00 06 04 04
    c_midich:           SeqSpec FF 7F 05 24 24 00 02 06

Standard MIDI provides for the port and channel specifications, but
they are apparently considered obsolete:

Obsolete meta-event:                Replacement:

    MIDI port (buss):   FF 21 01 po     Device (port) name: FF 09 len text
    MIDI channel:       FF 20 01 ch

What do other applications use for specifying port/channel?
```

7.19 seq64::mutex Class Reference

The mutex class provides a simple wrapper for the pthread_mutex_t type used as a recursive mutex.

Inheritance diagram for seq64::mutex:



Public Member Functions

- mutex ()

  The constructor assigns the recursive mutex to the local locking mutex.
- void lock () const

  Lock the mutex.
- void unlock () const

  Unlock the mutex.

Protected Attributes

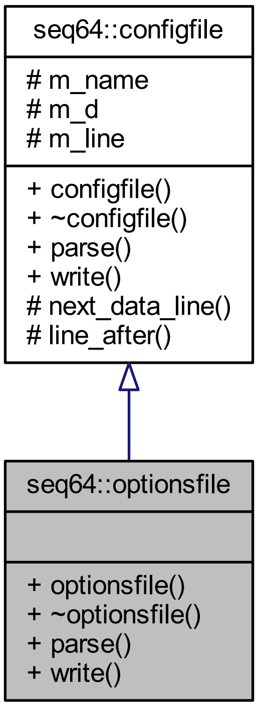
- pthread_mutex_t m_mutex_lock

  Provides a mutex lock usable by a single module or class.

7.20 seq64::optionsfile Class Reference

Provides a file for reading and writing the application' main configuration file.

Inheritance diagram for seq64::optionsfile:



Public Member Functions

- optionsfile (const std::string &name)

  Principal constructor.

- ∼optionsfile ()

  A rote destructor.
- bool parse (perform &perf)

  Parse the ∼*/.seq24rc or* ∼*/.config/sequencer64/sequencer64.rc file.*
- bool write (const perform &perf)

  This options-writing function is just about as complex as the options-reading function.

Additional Inherited Members

7.20.1 Detailed Description

The settings that are passed around are provided or used by the perform class.

7.20.2 Member Function Documentation

7.20.2.1 bool seq64::optionsfile::parse (perform & *a_perf*) [virtual]

[midi-control]

Get the number of sequence definitions provided in the [midi-control] section. Ranges from 32 on up. Then read in all of the sequence lines. The first 32 apply to the first screen set. There can also be a comment line "# mute in group" followed by 32 more lines. Then there are addditional comments and single lines for BPM up, BPM down, Screen Set Up, Screen Set Down, Mod Replace, Mod Snapshot, Mod Queue, Mod Gmute, Mod Glearn, and Screen Set Play. These are all forms of MIDI automation useful to control the playback while not sitting near the computer.

[mute-group]

The mute-group starts with a line that indicates up to 32 mute-groups are defined. A common value is 1024, which means there are 32 groups times 32 keys. But this value is currently thrown away. This value is followed by 32 lines of data, each contained 4 sets of 8 settings. See the seq24-doc project on GitHub for a much more detailed description of this section.

[midi-clock]

The MIDI-clock section defines the clocking value for up to 16 output busses. The first number, 16, indicates how many busses are specified. Generally, these busses are shown to the user with names such as "[1] seq24 1".

[keyboard-control]

The keyboard control defines the keys that will toggle the stage of each of up to 32 patterns in a pattern/sequence box. These keys are displayed in each box as a reminder. The first number specifies the Key number, and the second number specifies the Sequence number.

[keyboard-group]

The keyboard group specifies more automation for the application. The first number specifies the Key number, and the second number specifies the Group number. This section should be better described in the seq24-doc project on GitHub.

[jack-transport]

This section covers various JACK settings, one setting per line. In order, the following numbers are specfied:

```
-   jack_transport - Enable sync with JACK Transport.
-   jack_master - Seq24 will attempt to serve as JACK Master.
-   jack_master_cond - Seq24 will fail to be Master if there is
    already a Master set.
-   jack_start_mode:
    -   0 = Playback will be in Live mode.  Use this to allow
        muting and unmuting of loops.
    -   1 = Playback will use the Song Editor's data.
```

[midi-input]

This section covers the MIDI input busses, and has a format similar to "[midi-clock]". Generally, these busses are shown to the user with names such as "[1] seq24 1", and currently there is only one input buss. The first field is the port number, and the second number indicates whether it is disabled (0), or enabled (1).

[midi-clock-mod-ticks]

This section covers.... One common value is 64.

[manual-alsa-ports]

This section covers.... Set to 1 if you want seq24 to create its own ALSA ports and not connect to other clients.

[last-used-dir]

This section simply holds the last path-name that was used to read or write a MIDI file. We still need to add a check for a valid path, and currently the path must start with a "/", so it is not suitable for Windows.

[interaction-method]

This section specified the kind of mouse interaction.

- 0 = 'seq24' (original Seq24 method).
- 1 = 'fruity' (similar to a certain fruity sequencer we like).

The second data line is set to "1" if Mod4 can be used to keep seq24 in note-adding mode even after the right-click is released, and "0" otherwise.

Implements seq64::configfile.

7.20.2.2 bool seq64::optionsfile::write (const perform & *a_perf*) [virtual]

Parameters

a_perf	Provides a const reference to the main perform object. However, we have to cast away the constness, because too many of the perform getter functions are used in non-const contexts.

Returns

Returns true if the write operations all succeeded.

Implements seq64::configfile.

7.21 seq64::perform Class Reference

This class supports the performance mode.

Public Member Functions

- perform (gui_assistant &mygui)

  This construction initializes a vast number of member variables, some of them public (but we're working on that)!
- ∼perform ()

  The destructor sets some running flags to false, signals this condition, then joins the input and output threads if the were launched.
- const gui_assistant & gui () const

  'Getter' function for member m_gui_support The const getter.
- gui_assistant & gui ()

  'Getter' function for member m_gui_support The un-const getter.
- const keys_perform & keys () const

'Getter' function for member m_gui_support.keys() The const getter.

- keys_perform & keys ()

  'Getter' function for member m_gui_support.keys() The un-const getter.
- mastermidibus & master_bus ()

  'Getter' function for member m_master_bus
- bool is_running () const

  'Getter' function for member m_running
- bool is_learn_mode () const

  'Getter' function for member m_mode_group_learn
- void enregister (performcallback ∗pfcb)

  Adds a pointer to an object to be notified by this perform object.
- void init ()

  Initializes the master MIDI bus.
- void clear_all ()

  Clears all of the patterns/sequences.
- void launch_input_thread ()

  Creates the input thread using input_thread_func().
- void launch_output_thread ()

  Creates the output thread using output_thread_func().
- void init_jack ()

  Initializes JACK support, if SEQ64_JACK_SUPPORT is defined.
- void deinit_jack ()

  Tears down the JACK infrastructure.
- void add_sequence (sequence ∗a_seq, int a_perf)

  Adds a pattern/sequence pointer to the list of patterns.
- void delete_sequence (int a_num)

  Deletes a pattern/sequence by number.
- bool is_sequence_in_edit (int a_num)

  Check if the pattern/sequence, given by number, has an edit in progress.
- void clear_sequence_triggers (int a_seq)

  Clears the patterns/sequence for the given sequence, if it is active.
- bool is_sequence_valid (int a_sequence) const

  Provides common code to check for the bounds of a sequence number.
- bool is_sequence_invalid (int a_sequence) const

  Provides common code to check for the bounds of a sequence number.
- void set_left_tick (long a_tick)

  Set the left marker at the given tick.
- long get_left_tick () const

  'Getter' function for member m_left_tick
- void set_starting_tick (long a_tick)

  'Setter' function for member m_starting_tick
- long get_starting_tick () const

  'Getter' function for member m_starting_tick
- void set_right_tick (long a_tick)

  Set the right marker at the given tick.
- long get_right_tick () const

  'Getter' function for member m_right_tick
- void move_triggers (bool a_direction)

  If the left tick is less than the right tick, then, for each sequence that is active, its triggers are moved by the difference between the right and left in the specified direction.
- void copy_triggers ()

If the left tick is less than the right tick, then, for each sequence that is active, its triggers are copied, offset by the difference between the right and left.

- void push_trigger_undo ()

  For every active sequence, call that sequence's push_trigger_undo() function.
- void pop_trigger_undo ()

  For every active sequence, call that sequence's pop_trigger_undo() function.
- void print ()

  An information printing function with its body commented out.
- midi_control ∗ get_midi_control_toggle (unsigned int a_seq)

  Retrieves a value from m_midi_cc_toggle[].
- midi_control ∗ get_midi_control_on (unsigned int a_seq)

  Retrieves a value from m_midi_cc_on[].
- midi_control ∗ get_midi_control_off (unsigned int a_seq)

  Retrieves a value from m_midi_cc_off[].
- void handle_midi_control (int a_control, bool a_state)

  Handle the MIDI Control values that provide some automation for the application.
- const std::string & get_screen_set_notepad (int a_screen_set) const

  Retrieves the given string from m_screen_set_notepad[].
- const std::string & current_screen_set_notepad () const

  Returns the notepad text for the current screen-set.
- void set_screen_set_notepad (int screenset, const std::string &note)

  Copies the given string into m_screen_set_notepad[].
- void set_current_screen_set_notepad (const std::string &note)

  Sets the notepad text for the current screen-set.
- void set_screenset (int a_ss)

  Sets the m_screen_set value (the index or ID of the current screen set).
- int get_screenset () const

  'Getter' function for member m_screen_set
- void set_playing_screenset ()

  Sets the screen set that is active, based on the value of m_playing_screen.
- int get_playing_screenset () const

  'Getter' function for member m_playing_screen
- void mute_group_tracks ()

  Will need to study this one more closely.
- void select_and_mute_group (int a_g_group)

  Select a mute group and then mutes the track in the group.
- void set_mode_group_mute ()

  'Setter' function for member m_mode_group
- void unset_mode_group_mute ()

  'Setter' function for member m_mode_group Unsets this member.
- void select_group_mute (int a_g_mute)

  Makes some checks and sets the group mute flag.
- void set_mode_group_learn ()

  Sets the group-mute mode, then the group-learn mode, then notifies all of the notification subscribers.
- void unset_mode_group_learn ()

  Notifies all of the notification subscribers that group-learn is being turned off.
- void select_mute_group (int a_group)

  Will need to study this one more closely.
- void start (bool a_state)

  If JACK is not running, call inner_start() with the given state.
- void stop ()

If JACK is not running, call inner_stop().

- void start_jack ()

  If JACK is supported, starts the JACK transport.
- void stop_jack ()

  If JACK is supported, stops the JACK transport.
- void position_jack (bool a_state)

  If JACK is supported and running, sets the position of the transport.
- void off_sequences ()

  For all active patterns/sequences, set the playing state to false.
- void all_notes_off ()

  For all active patterns/sequences, turn off its playing notes.
- void set_active (int a_sequence, bool a_active)

  Sets or unsets the active state of the given pattern/sequence number.
- void set_was_active (int a_sequence)

  Sets was-active flags: main, edit, perf, and names.
- bool is_active (int a_sequence)

  Checks the pattern/sequence for activity.
- bool is_dirty_main (int a_sequence)

  Checks the pattern/sequence for main-dirtiness.
- bool is_dirty_edit (int a_sequence)

  Checks the pattern/sequence for edit-dirtiness.
- bool is_dirty_perf (int a_sequence)

  Checks the pattern/sequence for perf-dirtiness.
- bool is_dirty_names (int a_sequence)

  Checks the pattern/sequence for names-dirtiness.
- void new_sequence (int a_sequence)

  Creates a new pattern/sequence for the given slot, and sets the new pattern's master MIDI bus address.
- sequence $*$ get_sequence (int a_sequence)

  Retrieves the actual sequence, based on the pattern/sequence number.
- void reset_sequences ()

  For all active patterns/sequences, get its playing state, turn off the playing notes, set playing to false, zero the markers, and, if not in playback mode, restore the playing state.
- void play (long a_tick)

  Plays all notes to the current tick.
- void set_orig_ticks (long a_tick)

  For every pattern/sequence that is active, sets the "original ticks" value for the pattern.
- void set_bpm (int a_bpm)

  Sets the value of the BPM into the master MIDI buss, after making sure it is squelched to be between 20 and 500.
- int get_bpm ()

  Retrieves the BPM setting of the master MIDI buss.
- void set_looping (bool a_looping)

  'Setter' function for member m_looping
- void set_sequence_control_status (int a_status)

  If the given status is present in the c_status_snapshot, the playing state is saved.
- void unset_sequence_control_status (int a_status)

  If the given status is present in the c_status_snapshot, the playing state is restored.
- void set_group_mute_state (int a_g_track, bool a_mute_state)

  'Setter' function for member m_mute_group
- bool get_group_mute_state (int a_g_track)

  'Getter' function for member m_mute_group
- void mute_all_tracks ()

Mutes all tracks in the current set of active patterns/sequences.
- void output_func ()

  Performance output function.
- void input_func ()

  This function is called by input_thread_func().
- long get_max_trigger ()

  Locates the largest trigger value among the active sequences.
- void set_offset (int a_offset)

  Calculates the offset into the screen sets.
- void save_playing_state ()

  For all active patterns/sequences, this function gets the playing status and saves it in m_sequence_state[i].
- void restore_playing_state ()

  For all active patterns/sequences, this function gets the playing status from m_sequence_state[i] and sets it for the sequence.
- bool show_ui_sequence_key () const

  ***Accessor** m_show_ui_sequency_key*
- void start_playing (bool flag=false)

  Encapsulates a series of calls used in mainwnd.
- void stop_playing ()

  Encapsulates a series of calls used in mainwnd.
- void learn_toggle ()

  Encapsulates some calls used in mainwnd.
- int decrement_bpm ()

  Encapsulates some calls used in mainwnd.
- int increment_bpm ()

  Encapsulates some calls used in mainwnd.
- int decrement_screenset ()

  Encapsulates some calls used in mainwnd.
- int increment_screenset ()

  Encapsulates some calls used in mainwnd.
- void sequence_key (int seq)

  Handle a sequence key to toggle the playing of an active pattern in the selected screen-set.
- void set_input_bus (int bus, bool input_active)

  Sets the input bus, and handles the special "key-labels-on-sequence" functionality.
- bool mainwnd_key_event (const keystroke &k)

  Provided for mainwnd::on_key_press_event() and mainwnd::on_key_release_event() to call.
- bool perfroll_key_event (const keystroke &k, int drop_sequence)

  Provided for perfroll::on_key_press_event() and perfroll::on_key_release_event() to call.

7.21.1 Detailed Description

It has way too many data members, many of the public. Might be ripe for refactoring.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 seq64::perform::perform (gui_assistant & *mygui*)

Parameters

mygui	Provides access to the GUI assistant that holds many things, including the containers of keys and the "events" they provide. This is a base-class reference; for a real class, see the gui_assistant_gtk2 class in the seq_gtkmm2 GUI-specific library. Note that we access the m_gui_support member using the gui() accessor function.

7.21.2.2 seq64::perform::∼perform ( )

Finally, any active patterns/sequences are deleted.

7.21.3 Member Function Documentation

7.21.3.1 void seq64::perform::init ( )

Who calls this routine?

7.21.3.2 void seq64::perform::launch_input_thread ( )

This might be a good candidate for a small thread class derived from a small base class.

7.21.3.3 void seq64::perform::launch_output_thread ( )

This might be a good candidate for a small thread class derived from a small base class.

7.21.3.4 void seq64::perform::init_jack ( )

Who calls this routine?

7.21.3.5 void seq64::perform::add_sequence (sequence ∗ *a_seq,* int *a_perf*)

No check is made for a null pointer.

Check for preferred. This occurs if a_perf is in the valid range (0 to m_sequence_max) and it is not active. If preferred, then add it and activate it.

Otherwise, iterate through all patterns from a_perf to m_sequence_max and add and activate the first one that is not active, and then quit.

Warning

> The logic of the if-statement in this function was such that *a_perf* could be out-of-bounds in the else-clause. We reworked the logic to be airtight. This bug was caught by gcc 4.8.3 on CentOS, but not on gcc 4.9.3 on Debian Sid!

Parameters

a_seq	The number or index of the pattern/sequence to add.
a_perf	The performance number of the pattern? If this value is out-of-range, then it is ignored.

7.21.3.6 void seq64::perform::clear_sequence_triggers (int *a_seq*)

Parameters

a_seq	Provides the desired sequence. Hopefull, the is_active() function validates this value.

7.21.3.7 bool seq64::perform::is_sequence_valid (int *a_sequence*) const [inline]

Returns

Returns true if the sequence number is valid.

7.21.3.8 bool seq64::perform::is_sequence_invalid (int *a_sequence*) const [inline]

Returns

Returns true if the sequence number is invalid.

7.21.3.9 void seq64::perform::move_triggers (bool *a_direction*)

Parameters

a_direction	Specifies the desired direction; false = left, true = right.

7.21.3.10 void seq64::perform::copy_triggers ( )

This copies the triggers between the L marker and R marker to the R marker.

7.21.3.11 midi_control ∗ seq64::perform::get_midi_control_toggle (unsigned int *a_seq*)

Parameters

a_seq	Provides a control value (such as c_midi_control_bpm_up) to use to retrieve the desired midi_control object. Note that this value is unsigned simply to make the legality check of the parameter easier.

7.21.3.12 midi_control ∗ seq64::perform::get_midi_control_on (unsigned int *a_seq*)

Parameters

a_seq	Provides a control value (such as c_midi_control_bpm_up) to use to retrieve the desired midi_control object.

7.21.3.13 midi_control ∗ seq64::perform::get_midi_control_off (unsigned int *a_seq*)

Parameters

a_seq	Provides a control value (such as c_midi_control_bpm_up) to use to retrieve the desired midi_control object.

7.21.3.14 const std::string & seq64::perform::get_screen_set_notepad (int *screenset*) const

Parameters

screenset	The ID number of the string set, an index into the m_screen_set_notepad[] array. This value is validated.

Returns

Returns a reference to the desired string, or to an empty string if the screen-set number is invalid.

7.21.3.15 void seq64::perform::set_screen_set_notepad (int *screenset,* const std::string & *notepad*)

Parameters

screenset	The ID number of the string set, an index into the m_screen_set_xxx[] arrays.
notepad	Provides the string date to copy into the notepad. Not sure why a pointer is used, instead of nice "const std::string &" parameter. And this pointer isn't checked.

7.21.3.16 void seq64::perform::set_screenset (int *a_ss*)

Parameters

a_ss	The index of the desired string set. It is forced to range from 0 to c_max_sets - 1.

7.21.3.17 void seq64::perform::set_playing_screenset ( )

For each value up to c_seqs_in_set (32), the index of the current sequence in the currently screen set (m_playing↩_screen) is obtained. If it is active and the sequence actually exists

Modifies m_playing_screen, and mutes the group tracks.

7.21.3.18 void seq64::perform::unset_mode_group_learn ( )

Then unsets the group-learn mode flag..

7.21.3.19 void seq64::perform::select_mute_group (int *a_group*)

Parameters

a_group	Provides the group to mute. Note that this parameter is essentially a track or sequence number.

7.21.3.20 void seq64::perform::start (bool *a_state*)

Parameters

a_state	What does this state mean?

7.21.3.21 void seq64::perform::stop ( )

The logic seems backward here, in that we call inner_stop() if JACK is not running. Or perhaps we misunderstand the meaning of m_jack_running?

7.21.3.22 void seq64::perform::position_jack (bool *a_state*)

Warning

A lot of this code is effectively disabled by an early return statement.

7.21.3.23 void seq64::perform::all_notes_off ( )

Then flush the MIDI buss.

7.21.3.24 void seq64::perform::set_was_active (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

7.21.3.25 bool seq64::perform::is_active (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the value of the active-flag, or false if the pattern was invalid.

7.21.3.26 bool seq64::perform::is_dirty_main (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-main flag value, before setting it to false. Returns false if the pattern was invalid.

7.21.3.27 bool seq64::perform::is_dirty_edit (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-edit flag value, before setting it to false. Returns false if the pattern was invalid.

7.21.3.28 bool seq64::perform::is_dirty_perf (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-perf flag value, before setting it to false. Returns false if the pattern/sequence number was invalid.

7.21.3.29 bool seq64::perform::is_dirty_names (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-names flag value, before setting it to false. Returns false if the pattern/sequence number was invalid.

7.21.3.30 void seq64::perform::new_sequence (int *a_sequence*)

Then it activates the pattern.

It doesn't deal with thrown exceptions.

7.21.3.31 void seq64::perform::reset_sequences ( )

Then flush the MIDI buss.

7.21.3.32 void seq64::perform::play (long *a_tick*)

Starts the playing of all the patterns/sequences.

This function just runs down the list of sequences and has them dump their events.

Parameters

a_tick	Provides the tick at which to start playing.

7.21.3.33 void seq64::perform::set_orig_ticks (long *a_tick*)

Parameters

a_tick	

7.21.3.34 void seq64::perform::set_bpm (int *a_bpm*)

The value is set only if neither JACK nor this performance object are running.

Todo I think this logic is wrong, in that it needs only one of the two to be stopped before it sets the BPM, while it seems to me that both should be stopped; to be determined.

7.21.3.35 void seq64::perform::set_sequence_control_status (int *a_status*)

Then the given status is OR'd into the m_control_status.

7.21.3.36 void seq64::perform::unset_sequence_control_status (int *a_status*)

Then the given status is reversed in m_control_status.

7.21.3.37 void seq64::perform::output_func ( )

1. Get delta time (current - last).
2. Get delta ticks from time.
3. Add to current_ticks.
4. Compute prebuffer ticks.
5. Play from current tick to prebuffer.

Figure out how much time we need to sleep, and do it.

7.21.3.38 long seq64::perform::get_max_trigger ( )

Returns

Returns the highest trigger value, or zero. It is not clear why this function doesn't return a "no trigger found" value. Is there always at least one trigger, at 0?

7.21.3.39 void seq64::perform::set_offset (int *a_offset*) `[inline]`

Sets m_offset = a_offset ∗ c_mainwnd_rows ∗ c_mainwnd_cols;

Parameters

a_offset	The desired offset.

7.21.3.40 bool seq64::perform::show_ui_sequence_key ( ) const `[inline]`

Used in mainwid, options, optionsfile, userfile, and perform.

7.21.3.41 void seq64::perform::start_playing (bool *flag* = `false`) `[inline]`

We've reversed the start() and start_jack() calls so that JACK is started first, to match all of the other use-cases for playing that we've found in the code.

Todo Verify the usage and nature of this flag.

7.21.3.42 int seq64::perform::decrement_bpm ( ) `[inline]`

Actually does a lot of work in those function calls.

7.21.3.43 int seq64::perform::increment_bpm ( ) [inline]

Actually does a lot of work in those function calls.

7.21.3.44 void seq64::perform::set_input_bus (int *bus,* bool *input_active*)

This function is called by options::input_callback().

7.21.3.45 bool seq64::perform::mainwnd_key_event (const **keystroke** & *k*)

Returns

Returns true if the key was handled.

7.21.3.46 bool seq64::perform::perfroll_key_event (const **keystroke** & *k,* int *drop_sequence*)

Returns

Returns true if the key was handled.

7.22 seq64::performcallback Struct Reference

Provides for notification of events.

7.22.1 Detailed Description

Provide a response to a group-learn change event.

7.23 rc_settings Class Reference

This class contains the options formerly named "global_xxxxxx".

Public Member Functions

- rc_settings ()

  Default constructor.
- rc_settings (const rc_settings &rhs)

  Copy constructor.
- rc_settings & operator= (const rc_settings &rhs)

  Principal assignment operator.
- std::string home_config_directory () const

  Provides the directory for the configuration file, and also creates the directory if necessary.
- std::string config_filespec () const

  Constructs the full path and file specification for the "rc" file based on whether or not the legacy Seq24 filenames are being used.
- std::string user_filespec () const

  Constructs the full path and file specification for the "user" file based on whether or not the legacy Seq24 filenames are being used.
- void set_defaults ()

Sets the default values.
- void set_globals ()

  Copies the current values of the member variables into their corresponding global variables.
- void get_globals ()

  Copies the current values of the global variables into their corresponding member variables.
- bool legacy_format () const

  ***Accessor** m_legacy_format*
- bool lash_support () const

  ***Accessor** m_lash_support*
- bool allow_mod4_mode () const

  ***Accessor** m_allow_mod4_mode*
- bool show_midi () const

  ***Accessor** m_show_midi*
- bool priority () const

  ***Accessor** m_priority*
- bool stats () const

  ***Accessor** m_stats*
- bool pass_sysex () const

  ***Accessor** m_pass_sysex*
- bool with_jack_transport () const

  ***Accessor** m_with_jack_transport*
- bool with_jack_master () const

  ***Accessor** m_with_jack_master*
- bool with_jack_master_cond () const

  ***Accessor** m_with_jack_master_cond*
- bool jack_start_mode () const

  ***Accessor** m_jack_start_mode*
- bool manual_alsa_ports () const

  ***Accessor** m_manual_alsa_ports*
- bool is_pattern_playing () const

  ***Accessor** m_is_pattern_playing*
- bool print_keys () const

  ***Accessor** m_print_keys*
- bool device_ignore () const

  ***Accessor** m_device_ignore*
- int device_ignore_num () const

  'Getter' function for member m_device_ignore_num
- interaction_method_t interaction_method () const

  'Getter' function for member m_interaction_method
- const std::string & filename () const

  'Getter' function for member m_filename
- const std::string & jack_session_uuid () const

  'Getter' function for member m_jack_session_uuid
- const std::string & last_used_dir () const

  'Getter' function for member m_last_used_dir
- const std::string & config_directory () const

  'Getter' function for member m_config_directory
- const std::string & config_filename () const

  'Getter' function for member m_config_filename
- const std::string & user_filename () const

  'Getter' function for member m_user_filename

- const std::string & config_filename_alt () const

  'Getter' function for member m_config_filename_alt;
- const std::string & user_filename_alt () const

  'Getter' function for member m_user_filename_alt
- void device_ignore_num (int value)

  'Setter' function for member m_device_ignore_num However, please note that this value, while set in the options processing of the main module, does not appear to be used anywhere in the code in seq24, Sequencer24, and this application.
- void interaction_method (interaction_method_t value)

  'Setter' function for member m_interaction_method
- void filename (const std::string &value)

  'Setter' function for member m_filename
- void jack_session_uuid (const std::string &value)

  'Setter' function for member m_jack_session_uuid
- void last_used_dir (const std::string &value)

  'Setter' function for member m_last_used_dir
- void config_directory (const std::string &value)

  'Setter' function for member m_config_directory
- void config_filename (const std::string &value)

  'Setter' function for member m_config_filename
- void user_filename (const std::string &value)

  'Setter' function for member m_user_filename
- void config_filename_alt (const std::string &value)

  'Setter' function for member m_config_filename_alt;
- void user_filename_alt (const std::string &value)

  'Setter' function for member m_user_filename_alt

7.23.1 Member Function Documentation

7.23.1.1 std::string rc_settings::home_config_directory ( ) const

If the legacy format is in force, then the home directory for the configuration is (in Linux) "/home/username", and the configuration file is ".seq24rc".

If the new format is in force, then the home directory is (in Linux) "/home/username/.config/sequencer64", and the configuration file is "sequencer64.rc".

Returns

Returns the selection home configuration directory. If it does not exist orcould not be created, then an empty string is returned.

7.24 seq64::sequence Class Reference

The sequence class is firstly a receptable for a single track of MIDI data read from a MIDI file or edited into a pattern.

Public Types

- enum select_action_e {
  e_select ,
  e_deselect,
  e_toggle_selection,
  e_remove_one }

- typedef std::list< trigger > Triggers

  Exposes the triggers, currently needed for midi_container only.

Public Member Functions

- sequence ()

  Principal constructor.
- ∼sequence ()

  A rote destructor.
- sequence & operator= (const sequence &rhs)

  Principal assignment operator.
- event_list & events ()

  'Getter' function for member m_events
- Triggers & triggers ()

  'Getter' function for member m_triggers
- int event_count () const

  Returns the number of events stored in m_events.
- void push_undo ()

  Pushes the list-event into the undo-list.
- void pop_undo ()

  If there are items on the undo list, this function pushes the list-event into the redo-list, puts the top of the undo-list into the list-event, pops from the undo-list, calls verify_and_link(), and then calls unselect.
- void pop_redo ()

  If there are items on the redo list, this function pushes the list-event into the undo-list, puts the top of the redo-list into the list-event, pops from the redo-list, calls verify_and_link(), and then calls unselect.
- void push_trigger_undo ()

  Pushes the list-trigger into the trigger undo-list, then flags each item in the undo-list as unselected.
- void pop_trigger_undo ()

  If the trigger undo-list has any items, the list-trigger is pushed 9nto the redo list, the top of the undo-list is coped into the list-trigger, and then pops from the undo-list.
- void set_name (const std::string &name)

  Sets the sequence name member, m_name.
- void set_name (char ∗name)

  Sets the sequence name member, m_name.
- void set_bpm (long beats_per_measure)

  'Setter' function for member m_time_beats_per_measure
- long get_bpm () const

  'Getter' function for member m_time_beats_per_measure
- void set_bw (long beat_width)

  'Setter' function for member m_time_beat_width
- long get_bw () const

  'Getter' function for member m_time_beat_width
- void set_rec_vol (long rec_vol)

  'Setter' function for member m_rec_vol
- void set_song_mute (bool mute)

  'Setter' function for member m_song_mute
- bool get_song_mute () const

  'Getter' function for member m_song_mute
- const char ∗ get_name () const

  'Getter' function for member m_name pointer
- const std::string & name () const

'Getter' function for member m_name
- void set_editing (bool edit)

  'Setter' function for member m_editing
- bool get_editing () const

  'Getter' function for member m_editing
- void set_raise (bool edit)

  'Setter' function for member m_raise
- bool get_raise (void) const

  'Getter' function for member m_raise
- void set_length (long len, bool adjust_triggers=true)

  Sets the length (m_length) and adjusts triggers for it if desired.
- long get_length () const

  'Getter' function for member m_length
- long get_last_tick ()

  Returns the last tick played, and is used by the editor's idle function.
- void set_playing (bool)

  Sets the playing state of this sequence.
- bool get_playing () const

  'Getter' function for member m_playing
- void toggle_playing ()

  Toggles the playing status of this sequence.
- void toggle_queued ()

  'Setter' function for member m_queued and m_queued_tick
- void off_queued ()

  'Setter' function for member m_queued
- bool get_queued () const

  'Getter' function for member m_queued
- long get_queued_tick () const

  'Getter' function for member m_queued_tick
- void set_recording (bool)

  'Setter' function for member m_recording and m_notes_on
- bool get_recording () const

  'Getter' function for member m_recording
- void set_snap_tick (int st)

  'Setter' function for member m_snap_tick
- void set_quantized_rec (bool qr)

  'Setter' function for member m_quantized_rec
- bool get_quantized_rec () const

  'Getter' function for member m_quantized_rec
- void set_thru (bool)

  'Setter' function for member m_thru
- bool get_thru () const

  'Getter' function for member m_thru
- bool is_dirty_main ()

  Returns the value of the dirty main flag, and sets that flag to false (i.e.
- bool is_dirty_edit ()

  Returns the value of the dirty edit flag, and sets that flag to false.
- bool is_dirty_perf ()

  Returns the value of the dirty performance flag, and sets that flag to false.
- bool is_dirty_names ()

  Returns the value of the dirty names (heh heh) flag, and sets that flag to false.

- void set_dirty_mp ()

  Sets the dirty flags for names, main, and performance.
- void set_dirty ()

  Call set_dirty_mp() and then sets the dirty flag for editing.
- unsigned char get_midi_channel () const

  'Getter' function for member m_midi_channel
- void set_midi_channel (unsigned char ch)

  Sets the m_midi_channel number.
- void print ()

  Prints a list of the currently-held events.
- void print_triggers ()

  Prints a list of the currently-held triggers.
- void play (long tick, bool playback_mode)

  The play() function dumps notes starting from the given tick, and it pre-buffers ahead.
- void set_orig_tick (long tick)

  'Setter' function for member m_last_tick
- void add_event (const event ∗e)

  Adds an event to the internal event list in a sorted manner.
- void add_trigger (long tick, long length, long offset=0, bool adjust_offset=true)

  Adds a trigger.
- void split_trigger (long tick)

  Splits a trigger.
- void grow_trigger (long tick_from, long tick_to, long length)

  Grows a trigger.
- void del_trigger (long tick)

  Deletes a trigger, that brackets the given tick, from the trigger-list.
- bool unselect_triggers ()

  Always returns false!
- bool intersectTriggers (long position, long &start, long &end)

  This function examines each trigger in the trigger list.
- bool intersectNotes (long position, long position_note, long &start, long &end, long &note)

  This function examines each note in the event list.
- bool intersectEvents (long posstart, long posend, long status, long &start)

  This function examines each non-note event in the event list.
- void move_selected_triggers_to (long tick, bool adjust_offset, int which=2)

  Moves selected triggers as per the given parameters.
- long get_selected_trigger_start_tick ()

  Gets the selected trigger's start tick.
- long get_selected_trigger_end_tick ()

  Gets the selected trigger's end tick.
- long get_max_trigger ()

  Get the ending value of the last trigger in the trigger-list.
- void move_triggers (long start_tick, long distance, bool direction)

  Moves triggers in the trigger-list.
- void copy_triggers (long start_tick, long distance)

  Not sure what these diagrams are for yet.
- void clear_triggers ()

  Clears the whole list of triggers.
- long get_trigger_offset () const

  'Getter' function for member m_trigger_offset
- void set_midi_bus (char mb)

Sets the midibus number to dump to.
- char get_midi_bus () const

  'Getter' function for member m_bus
- void set_master_midi_bus (mastermidibus ∗mmb)

  'Setter' function for member m_masterbus
- int select_note_events (long tick_s, int note_h, long tick_f, int note_l, select_action_e action)

  This function selects events in range of tick start, note high, tick end, and note low.
- int select_events (long tick_s, long tick_f, unsigned char status, unsigned char cc, select_action_e action)

  Select all events in the given range, and returns the number selected.
- int select_events (unsigned char status, unsigned char cc, bool inverse=false)

  Select all events with the given status, and returns the number selected.
- int get_num_selected_notes ()

  Counts the selected notes in the event list.
- int get_num_selected_events (unsigned char status, unsigned char cc)

  Counts the selected events, with the given status, in the event list.
- void select_all ()

  Selects all events, unconditionally.
- void copy_selected ()

  Copies the selected events.
- void paste_selected (long tick, int note)

  Pastes the selected notes (and only note events) at the given tick and the given note value.
- void get_selected_box (long &tick_s, int &note_h, long &tick_f, int &note_l)

  Returns the 'box' of the selected items.
- void get_clipboard_box (long &tick_s, int &note_h, long &tick_f, int &note_l)

  Returns the 'box' of selected items.
- void move_selected_notes (long delta_tick, int delta_note)

  Removes and adds reads selected in position.
- void add_note (long tick, long length, int note, bool paint=false)

  Adds a note of a given length and note value, at a given tick location.
- void add_event (long tick, unsigned char status, unsigned char d0, unsigned char d1, bool paint=false)

  Adds a event of a given status value and data values, at a given tick location.
- void stream_event (event ∗ev)

  Streams the given event.
- void change_event_data_range (long tick_s, long tick_f, unsigned char status, unsigned char cc, int d_s, int d_f)

  Changes the event data range.
- void increment_selected (unsigned char status, unsigned char control)

  Increments events the match the given status and control values.
- void decrement_selected (unsigned char status, unsigned char control)

  Decrements events the match the given status and control values.
- void grow_selected (long delta_tick)

  Moves note off event.
- void stretch_selected (long delta_tick)

  Performs a stretch operation on the selected events.
- void remove_marked ()

  Removes marked events.
- void mark_selected ()

  Marks the selected events.
- void unpaint_all ()

  Unpaints all list-events.
- void unselect ()

Deselects all events, unconditionally.

- void verify_and_link ()

  This function verifies state: all note-ons have an off, and it links note-offs with their note-ons.
- void link_new ()

  Links a new event.
- void zero_markers ()

  Resets everything to zero.
- void play_note_on (int note)

  Plays a note from the piano roll on the main bus on the master MIDI buss.
- void play_note_off (int note)

  Turns off a note from the piano roll on the main bus on the master MIDI buss.
- void off_playing_notes ()

  Sends a note-off event for all active notes.
- void reset_draw_marker ()

  This refreshes the play marker to the last tick.
- void reset_draw_trigger_marker ()

  Threadsafe
- draw_type get_next_note_event (long ∗tick_s, long ∗tick_f, int ∗note, bool ∗selected, int ∗velocity)

  Each call to seqdata() fills the passed references with a events elements, and returns true.
- int get_lowest_note_event ()

  Threadsafe
- int get_highest_note_event ()

  Threadsafe
- bool get_next_event (unsigned char status, unsigned char cc, long ∗tick, unsigned char ∗d0, unsigned char ∗d1, bool ∗selected)

  Get the next event in the event list that matches the given status and control character.
- bool get_next_event (unsigned char ∗status, unsigned char ∗cc)

  Get the next event in the event list.
- bool get_next_trigger (long ∗tick_on, long ∗tick_off, bool ∗selected, long ∗tick_offset)

  Get the next trigger in the trigger list, and set the parameters based on that trigger.
- void fill_container (midi_container &c, int tracknumber)

  This function fills the given character list with MIDI data from the current sequence, preparatory to writing it to a file.
- void transpose_notes (int steps, int scale)

  Transposes notes by the given steps, in accordance with the given scale.

7.24.1 Detailed Description

More members than you can shake a stick at.

7.24.2 Member Enumeration Documentation

7.24.2.1 enum seq64::sequence::select_action_e

Enumerator

e_select This enumeration is used in selecting events and note. Se the select_note_events() and select_↩ events() functions.

e_deselect To deselect the event under the cursor.

e_toggle_selection To toggle the selection of the event under the cursor.

e_remove_one To remove one note under the cursor.

7.24.3 Member Function Documentation

7.24.3.1 sequence & seq64::sequence::operator= (const sequence & *rhs*)

Follows the stock rules for such an operator, but does a little more then just assign member values.

Threadsafe

7.24.3.2 int seq64::sequence::event_count ( ) const

Threadsafe

7.24.3.3 void seq64::sequence::push_undo ( )

Threadsafe

7.24.3.4 void seq64::sequence::pop_undo ( )

Threadsafe

7.24.3.5 void seq64::sequence::pop_redo ( )

Threadsafe

7.24.3.6 void seq64::sequence::push_trigger_undo ( )

Threadsafe

7.24.3.7 void seq64::sequence::set_bpm (long *beats_per_measure*)

Threadsafe

7.24.3.8 void seq64::sequence::set_bw (long *beat_width*)

Threadsafe

7.24.3.9 long seq64::sequence::get_bw ( ) const `[inline]`

Threadsafe

7.24.3.10 void seq64::sequence::set_rec_vol (long *rec_vol*)

Threadsafe

7.24.3.11 void seq64::sequence::set_length (long *len,* bool *adjust_triggers* = `true`)

Threadsafe

7.24.3.12 void seq64::sequence::set_playing (bool *a_p*)

When playing, and the sequencer is running, notes get dumped to the ALSA buffers.

Parameters

a_p	Provides the playing status to set. True means to turn on the playing, false means to turn it off, and turn off any notes still playing.

7.24.3.13 void seq64::sequence::toggle_queued ( )

Toggles the queued flag and sets the dirty-mp flag. Also calculates the queued tick based on m_last_tick.

Threadsafe

7.24.3.14 void seq64::sequence::off_queued ( )

Toggles the queued flag and sets the dirty-mp flag.

Threadsafe

7.24.3.15 void seq64::sequence::set_recording (bool *a_r*)

Threadsafe

7.24.3.16 void seq64::sequence::set_snap_tick (int *a_st*)

Threadsafe

7.24.3.17 void seq64::sequence::set_quantized_rec (bool *a_qr*)

Threadsafe

7.24.3.18 void seq64::sequence::set_thru (bool *a_r*)

Threadsafe

7.24.3.19 bool seq64::sequence::is_dirty_main ( )

resets it). This flag signals that a redraw is needed from recording.

Threadsafe

7.24.3.20 bool seq64::sequence::is_dirty_edit ( )

Threadsafe

7.24.3.21 bool seq64::sequence::is_dirty_perf ( )

Threadsafe

7.24.3.22 bool seq64::sequence::is_dirty_names ( )

Threadsafe

7.24.3.23 void seq64::sequence::set_dirty_mp ( )

Not threadsafe

7.24.3.24 void seq64::sequence::set_dirty ( )

Threadsafe

7.24.3.25 void seq64::sequence::set_midi_channel (unsigned char *a_ch*)

Threadsafe

7.24.3.26 void seq64::sequence::print ( )

Not threadsafe

7.24.3.27 void seq64::sequence::print_triggers ( )

Not threadsafe

7.24.3.28 void seq64::sequence::play (long *tick*, bool *playback_mode*)

This function is called by the sequencer thread, performance. The tick comes in as global tick.

It turns the sequence off after we play in this frame.

Threadsafe

7.24.3.29 void seq64::sequence::set_orig_tick (long *tick*)

Threadsafe

7.24.3.30 void seq64::sequence::add_event (const event $*$ *ep*)

Then it reset the draw-marker and sets the dirty flag.

Currently, when reading a MIDI file (see the midifile module's parse function), only the main events (notes, after-touch, pitch, program changes, etc.) are added with this function. So, we can rely on reading only playable events into a sequence.

This module (sequencer) adds all of those events as well, but it can surely add other events. We should assume that any events added by sequencer are playable.

Threadsafe

Warning

> This pushing (and, in writing the MIDI file, the popping), causes events with identical timestamps to be written in reverse order. Doesn't affect functionality, but it's puzzling until one understands what is happening.

7.24.3.31 void seq64::sequence::add_trigger (long *a_tick*, long *a_length*, long *a_offset* = `0`, bool *a_adjust_offset* = `true`)

If a_state = true, the range is on. If a_state = false, the range is off.

What is this?

```
is      ie
<      ><        ><        >
es             ee
<               >
XX

es ee
<   >
<>

es    ee
<      >
<    >

es     ee
<       >
<    >
```

7.24.3.32 void seq64::sequence::split_trigger (long *a_tick*)

This is the public overload of split_trigger.

Threadsafe

7.24.3.33 void seq64::sequence::grow_trigger (long *a_tick_from,* long *a_tick_to,* long *a_length*)

Threadsafe

7.24.3.34 void seq64::sequence::del_trigger (long *a_tick*)

Threadsafe

7.24.3.35 bool seq64::sequence::intersectTriggers (long *position,* long & *start,* long & *end*)

If the given position is between the current trigger's tick-start and tick-end values, the these values are copied to the start and end parameters, respectively, and then we exit.

Threadsafe

Parameters

position	The position to examine.
start	The destination for the starting tick (m_tick_start) of the matching trigger.
end	The destination for the ending tick (m_tick_end) of the matching trigger.

Returns

Returns true if a trigger was found whose start/end ticks contained the position. Otherwise, false is returned, and the start and end return parameters should not be used.

7.24.3.36 bool seq64::sequence::intersectNotes (long *position,* long *position_note,* long & *start,* long & *ender,* long & *note*)

If the given position is between the current notes on and off time values, values, the these values are copied to the start and end parameters, respectively, the note value is copied to the note parameter, and then we exit.

Threadsafe

Parameters

position	The position to examine.
position_note	I think this is the note value we might be looking for ???
start	The destination for the starting tick (m_tick_start) of the matching trigger.
end	The destination for the ending tick (m_tick_end) of the matching trigger.
note	The destination for the note of the matching event.

Returns

Returns true if a event was found whose start/end ticks contained the position. Otherwise, false is returned, and the start and end return parameters should not be used.

7.24.3.37 bool seq64::sequence::intersectEvents (long *posstart,* long *posend,* long *status,* long & *start*)

If the given position is between the current trigger's tick-start and tick-end values, the these values are copied to the start and end parameters, respectively, and then we exit.

Threadsafe

Parameters

posstart	The starting position to examine.
posend	The ending position to examine.
status	The desired status value.
start	The destination for the starting tick (m_tick_start) of the matching trigger.

Returns

Returns true if a event was found whose start/end ticks contained the position. Otherwise, false is returned, and the start and end return parameters should not be used.

7.24.3.38 void seq64::sequence::move_selected_triggers_to (long *a_tick,* bool *a_adjust_offset,* int *a_which* = 2)

```
          min_tick][0                1][max_tick
                           2
```

```
-   If we are moving the 0, use first as offset.
-   If we are moving the 1, use the last as the offset.
-   If we are moving both (2), use first as offset.
```

Threadsafe

7.24.3.39 long seq64::sequence::get_selected_trigger_start_tick ( )

Threadsafe

7.24.3.40 long seq64::sequence::get_selected_trigger_end_tick ( )

Threadsafe

7.24.3.41 long seq64::sequence::get_max_trigger ( )

Threadsafe

7.24.3.42 void seq64::sequence::move_triggers (long *a_start_tick,* long *a_distance,* bool *a_direction*)

Threadsafe

7.24.3.43 void seq64::sequence::copy_triggers (long *a_start_tick,* long *a_distance*)

```
... a
[      ][      ]
...
... a
...

5   7    play
3        offset
8   10   play

X...X...X...X...X...X...X...X...X...X...
L       R
[        ] [     ]  []  orig
[                    ]

        <<
        [     ]    [  ][ ]  [] split on the R marker, shift first
        [     ]        [     ]
        delete middle
        [     ][ ]  []         move ticks
        [     ][     ]

        L       R
        [     ][ ] [     ]  [] split on L
        [     ][             ]

        [     ]        [ ] [     ]  [] increase all after L
        [     ]        [             ]
```

Copies triggers to...

Threadsafe

7.24.3.44 void seq64::sequence::clear_triggers ( )

Threadsafe

7.24.3.45 void seq64::sequence::set_midi_bus (char *mb*)

Threadsafe

7.24.3.46 void seq64::sequence::set_master_midi_bus (mastermidibus $*$ *mmb*)

Threadsafe

7.24.3.47 int seq64::sequence::select_note_events (long *a_tick_s,* int *a_note_h,* long *a_tick_f,* int *a_note_l,* select_action_e *a_action*)

Returns the number selected.

Threadsafe

7.24.3.48 int seq64::sequence::select_events (long *tick_s*, long *tick_f*, unsigned char *status*, unsigned char *cc*, select_action_e *action*)

Note that there is also an overloaded version of this function.

Threadsafe

7.24.3.49 int seq64::sequence::select_events (unsigned char *status*, unsigned char *cc*, bool *inverse* = `false`)

Note that there is also an overloaded version of this function.

Threadsafe

Warning

This used to be a void function, so it just returns 0 for now.

7.24.3.50 int seq64::sequence::get_num_selected_notes ( )

Threadsafe

7.24.3.51 int seq64::sequence::get_num_selected_events (unsigned char *status*, unsigned char *cc*)

If the event is a control change (CC), then it must also match the given CC value.

Threadsafe

7.24.3.52 void seq64::sequence::select_all ( )

Threadsafe

7.24.3.53 void seq64::sequence::copy_selected ( )

Threadsafe

7.24.3.54 void seq64::sequence::paste_selected (long *tick*, int *note*)

I wonder if we can get away with just getting a reference to m_events_clipboard, rather than copying the whole thing, for speed.

Threadsafe

7.24.3.55 void seq64::sequence::add_note (long *tick*, long *length*, int *note*, bool *paint* = `false`)

It adds a single note-on / note-off pair.

The a_paint parameter indicates if we care about the painted event, so then the function runs though the events and deletes the painted ones that overlap the ones we want to add.

Threadsafe

7.24.3.56 void seq64::sequence::add_event (long *a_tick*, unsigned char *a_status*, unsigned char *a_d0*, unsigned char *a_d1*, bool *a_paint* = `false`)

The a_paint parameter indicates if we care about the painted event, so then the function runs though the events and deletes the painted ones that overlap the ones we want to add.

Threadsafe

7.24.3.57 void seq64::sequence::stream_event (event ∗ *ev*)

Threadsafe

7.24.3.58 void seq64::sequence::change_event_data_range (long *tick_s*, long *tick_f*, unsigned char *status*, unsigned char *cc*, int *data_s*, int *data_f*)

Changes only selected events, if any.

Threadsafe

Let t == the current tick value; ts == tick start value; tf == tick finish value; ds = data start value; df == data finish value; d = the new data value.

Then

```
             df (t - ts) + ds (tf - t)
        d = --------------------------
                    tf  -   ts

If this were an interpolation formula it would be:

                            t -  ts
        d = ds + (df - ds) ---------
                            tf - ts

Something is not quite right; to be investigated.

\param tick_s
    Provides the starting tick value.

\param tick_f
    Provides the ending tick value.

\param status
    Provides the event status that is to be changed.

\param cc
    Provides the event control value.

\param data_s
    Provides the starting data value.

\param data_f
    Provides the finishing data value.
```

7.24.3.59 void seq64::sequence::increment_selected (unsigned char *astat*, unsigned char *control*)

The supported statuses are:

```
-   EVENT_NOTE_ON
-   EVENT_NOTE_OFF
-   EVENT_AFTERTOUCH
-   EVENT_CONTROL_CHANGE
-   EVENT_PITCH_WHEEL
-   EVENT_PROGRAM_CHANGE
-   EVENT_CHANNEL_PRESSURE
```

Threadsafe

7.24.3.60 void seq64::sequence::decrement_selected (unsigned char *astat*, unsigned char *control*)

The supported statuses are:

```
-    EVENT_NOTE_ON
-    EVENT_NOTE_OFF
-    EVENT_AFTERTOUCH
-    EVENT_CONTROL_CHANGE
-    EVENT_PITCH_WHEEL
-    EVENT_PROGRAM_CHANGE
-    EVENT_CHANNEL_PRESSURE
```

Threadsafe

7.24.3.61 void seq64::sequence::grow_selected (long *delta_tick*)

Threadsafe

7.24.3.62 void seq64::sequence::stretch_selected (long *delta_tick*)

This should move a note off event, according to old comments, but it doesn't seem to do that. See the grow_↩ selected() function.

Threadsafe

7.24.3.63 void seq64::sequence::remove_marked ( )

Note how this function handles removing a value to avoid incrementing a now-invalid iterator.

Threadsafe

7.24.3.64 void seq64::sequence::mark_selected ( )

Threadsafe

7.24.3.65 void seq64::sequence::unpaint_all ( )

Threadsafe

7.24.3.66 void seq64::sequence::unselect ( )

Threadsafe

7.24.3.67 void seq64::sequence::verify_and_link ( )

Threadsafe

7.24.3.68 void seq64::sequence::link_new ( )

Threadsafe

7.24.3.69 void seq64::sequence::zero_markers ( )

This function is used when the sequencer stops.

Threadsafe

7.24.3.70 void seq64::sequence::play_note_on (int *a_note*)

It flushes a note to the midibus to preview its sound, used by the virtual piano.

Threadsafe

7.24.3.71 void seq64::sequence::play_note_off (int *a_note*)

Threadsafe

7.24.3.72 void seq64::sequence::off_playing_notes ( )

Threadsafe

7.24.3.73 void seq64::sequence::reset_draw_marker ( )

It resets the draw marker so that calls to get_next_note_event() will start from the first event.

Threadsafe

7.24.3.74 draw_type seq64::sequence::get_next_note_event (long ∗ *a_tick_s,* long ∗ *a_tick_f,* int ∗ *a_note,* bool ∗ *a_selected,* int ∗ *a_velocity*)

When it has no more events, returns a false.

7.24.3.75 bool seq64::sequence::get_next_event (unsigned char *status,* unsigned char *cc,* long ∗ *tick,* unsigned char ∗ *d0,* unsigned char ∗ *d1,* bool ∗ *selected*)

Then set the rest of the parameters parameters using that event.

7.24.3.76 bool seq64::sequence::get_next_event (unsigned char ∗ *a_status,* unsigned char ∗ *a_cc*)

Then set the status and control character parameters using that event.

7.24.3.77 void seq64::sequence::fill_container (midi_container & *c,* int *tracknumber*)

Note that some of the events might not come out in the same order they were stored in (we see that with program-change events.

Parameters

c	Provides the std::list object to push events to the front, which thus inserts them in backwards order. (These events are then popped back, which restores the order, with some exceptions).

tracknumber	Provides the track number. This number is masked into the track information.

7.24.3.78 void seq64::sequence::transpose_notes (int *steps*, int *scale*)

If the scale value is 0, this is "no scale", which is the chromatic scale, where all 12 notes, including sharps and flats, are part of the scale.

7.25 seq64::trigger Class Reference

This class is used in playback.

Public Member Functions

- trigger ()

  Initializes the trigger structure.
- bool operator< (const trigger &rhs)

  This operator compares only the m_tick_start members.

7.25.1 Detailed Description

Making its members public makes it really "just" a structure.

7.26 user_instrument Class Reference

Provides data about the MIDI instruments, readable from the "user" configuration file.

Public Member Functions

- user_instrument (const std::string &name="")

  Default constructor.
- user_instrument (const user_instrument &rhs)

  Copy constructor.
- user_instrument & operator= (const user_instrument &rhs)

  Principal assignment operator.
- bool is_valid () const

  'Getter' function for member m_is_valid
- void set_defaults ()

  Sets the default values.
- void set_global (int instrum) const

  Copies the current values of the member variables into the selected legacy global variable.
- void get_global (int instrum)

  Copies the current values of the selected legacy global variable into corresponding member variable.
- const std::string & name () const

  'Getter' function for member m_instrument_def.instrument (name of instrument)
- int controller_count () const

  'Getter' function for member m_controller_count This function returns the number of active controllers.
- int controller_max () const

'Getter' function for member MIDI_CONTROLLER_MAX This function returns the maximum number of controllers, active or inactive.

- const std::string & controller_name (int c) const

  'Getter' function for member m_instrument_def.controllers[c]
- bool controller_active (int c) const

  'Getter' function for member m_instrument_def.controllers_active[c]
- void set_controller (int c, const std::string &cname, bool isactive)

  'Setter' function for member m_instrument_def.controllers[c] and .controllers_active[c] Only sets the controller values if the object is already valid.

7.26.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

7.26.2 Member Function Documentation

7.26.2.1 void user_instrument::set_defaults ( )

Also invalidates the object.

7.26.2.2 void user_instrument::set_global (int *instrum*) const

Should be called at initialization, and after settings are read from the "user" configuration file.

This function fills in all of the MIDI_CONTROLLER_MAX (128) values of the controllers and controllers_active fields.

Note that this is done only if the object is valid.

Parameters

instrum	Provides the destination instrument number. In order to support the legacy code, this index value must be less than c_max_instruments (64).

7.26.2.3 void user_instrument::get_global (int *instrum*)

Should be called before settings are written to the "user" configuration file.

This function fills in all of the MIDI_CONTROLLER_MAX (128) values of the controllers and controllers_active fields.

This function also sets the validity flag to true if the instrument name is not empty; the rest of the values are not checked.

Parameters

instrum	Provides the source instrument number. In order to support the legacy code, this index value must be less than c_max_instruments (64).

7.26.2.4 int user_instrument::controller_max ( ) const `[inline]`

Remember that the controller numbers for each MIDI instrument range from 0 to 127 (MIDI_CONTROLLER_MAX-1).

7.26.2.5 const std::string & user_instrument::controller_name (int *c*) const

Parameters

c	The index of the desired controller.

Returns

The name of the desired controller has is returned. If the index c is out of range, or the object is not valid, then a reference to an internal, empty string is returned.

7.26.2.6 bool user_instrument::controller_active (int *c*) const

Parameters

c	The index of the desired controller.

Returns

The status of the desired controller has is returned. If the index c is out of range, or the object is not valid, then false is returned.

7.26.2.7 void user_instrument::set_controller (int *c*, const std::string & *cname*, bool *isactive*)

Parameters

c	The index of the desired controller.
cname	The name of the controller to be set as the controller name.
isactive	A flag that indicates if the desired controller is active.

7.27 user_instrument_t Struct Reference

This structure corresponds to `[user-instrument-N]` definitions in the `~/.seq24usr` or `~/.config/sequencer64/s` `rc` file.

7.28 user_midi_bus Class Reference

Provides data about the MIDI busses, readable from the "user" configuration file.

Public Member Functions

- user_midi_bus (const std::string &name="")

  Default constructor.
- user_midi_bus (const user_midi_bus &rhs)

  Copy constructor.
- user_midi_bus & operator= (const user_midi_bus &rhs)

  Principal assignment operator.
- bool is_valid () const

  'Getter' function for member m_is_valid
- void set_defaults ()

  Sets the default values.
- void set_global (int buss) const

Copies the current values of the member variables into their corresponding global variables.

- void get_global (int buss)

    Copies the current values of the global variables into their corresponding member variable.

- const std::string & name () const

    'Getter' function for member m_midi_bus_def.alias (name of alias)

- int channel_count () const

    'Getter' function for member m_channel_count

- int channel_max () const

    'Getter' function for member MIDI_BUS_CHANNEL_MAX

- int instrument (int channel) const

    'Getter' function for member m_midi_bus_def.instrument[channel]

- void set_instrument (int channel, int instrum)

    'Getter' function for member m_midi_bus_def.instrument[channel]

7.28.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

7.28.2 Member Function Documentation

7.28.2.1 void user_midi_bus::set_defaults ( )

Also invalidates the object. All 16 of the channels are set to GM_INSTRUMENT_FLAG (-1).

7.28.2.2 void user_midi_bus::set_global (int *buss*) const

Should be called at initialization, and after settings are read from the "user" configuration file.

Note that this is done only if the object is valid.

Parameters

buss	Provides the destination buss number. In order to support the legacy code, this index value must be less than c_max_busses (32).

7.28.2.3 void user_midi_bus::get_global (int *buss*)

Should be called before settings are written to the "user" configuration file.

This function also sets the validity flag to true if the instrument name is not empty; the rest of the values are not checked.

Parameters

buss	Provides the destination buss number. In order to support the legacy code, this index value must be less than c_max_busses (32).

7.28.2.4 int user_midi_bus::channel_count ( ) const `[inline]`

Returns

This function returns the number of channels. Basically this value is always the same as that returned by channel_max(), but this pair of functions is consistent with the count functions in the user_instrument class.

7.28.2.5 int user_midi_bus::channel_max ( ) const [inline]

Returns

Returns the maximum number of MIDI buss channels. Remember that the instrument channels for each MIDI buss range from 0 to 15 (MIDI_BUS_CHANNEL_MAX-1).

7.28.2.6 int user_midi_bus::instrument (int *channel*) const

Parameters

channel	Provides the desired buss channel number.

Returns

The instrument number of the desired buss channel is returned. If the channel number is out of range, or the object is not valid, then GM_INSTRUMENT_FLAG (-1) is returned.

7.28.2.7 void user_midi_bus::set_instrument (int *channel,* int *instrum*)

Does not alter the validity flag, just checks it.

Parameters

channel	Provides the desired buss channel number.
instrum	Provides the instrument number to set that channel to.

7.29 user_midi_bus_t Struct Reference

This structure corresponds to `[user-midi-bus-0]` definitions in the `~/.seq24usr` ("user") file.

7.30 user_settings Class Reference

Holds the current values of sequence settings and settings that can modify the number of sequences and the configuration of the user-interface.

Public Member Functions

- user_settings ()

  Default constructor.
- user_settings (const user_settings &rhs)

  Copy constructor.
- user_settings & operator= (const user_settings &rhs)

  Principal assignment operator.
- void set_defaults ()

  Sets the default values.
- void normalize ()

  Calculate the derived values from the already-set values.
- void set_globals () const

  Copies the current values of the member variables into their corresponding global variables.

- void get_globals ()

  Copies the current values of the global variables into their corresponding member variables.
- bool add_bus (const std::string &alias)

  Adds a user bus to the container, but only does so if the name parameter is not empty.
- bool add_instrument (const std::string &instname)

  Adds a user instrument to the container, but only does so if the name parameter is not empty.
- const user_midi_bus & bus (int index)

  'Getter' function for member Unlike the non-const version this function is public.
- const user_instrument & instrument (int index)

  'Getter' function for member Unlike the non-const version this function is public.
- int bus_count () const

  'Getter' function for member m_midi_buses.size()
- void set_bus_instrument (int index, int channel, int instrum)

  'Getter' function for member m_midi_buses[index].instrument[channel] Currently this function is used, in the userfile↩ ::parse() function.
- int bus_instrument (int buss, int channel)

  'Getter' function for member m_midi_buses[buss].instrument[channel]
- const std::string & bus_name (int buss)

  'Getter' function for member m_midi_buses[buss].name
- int instrument_count () const

  'Getter' function for member m_instruments.size()
- void set_instrument_controllers (int index, int cc, const std::string &ccname, bool isactive)

  'Setter' function for member m_midi_instrument_defs[index].controllers, controllers_active
- const std::string & instrument_name (int instrum)

  'Getter' function for member m_instruments[instrument].instrument (name of instrument)
- bool instrument_controller_active (int instrum, int c)

  'Getter' function for member m_instruments[instrument].controllers_active[controller]
- const std::string & instrument_controller_name (int instrum, int c)

  'Getter' function for member m_instruments[instrument].controllers_active[controller]
- int mainwnd_rows () const

  'Getter' function for member m_mainwnd_rows
- int mainwnd_cols () const

  'Getter' function for member m_mainwnd_cols
- int seqs_in_set () const

  'Getter' function for member m_seqs_in_set
- int gmute_tracks () const

  'Getter' function for member m_gmute_tracks
- int max_sets () const

  'Getter' function for member m_max_sets
- int max_sequence () const

  'Getter' function for member m_max_sequence
- int text_x () const

  'Getter' function for member m_text_x
- int text_y () const

  'Getter' function for member m_text_y
- int seqchars_x () const

  'Getter' function for member m_seqchars_x
- int seqchars_y () const

  'Getter' function for member m_seqchars_y
- int seqarea_x () const

  'Getter' function for member m_seqarea_x

- int seqarea_y () const

  'Getter' function for member m_seqarea_y
- int seqarea_seq_x () const

  'Getter' function for member m_seqarea_seq_x
- int seqarea_seq_y () const

  'Getter' function for member m_seqarea_seq_y
- int mainwid_border () const

  'Getter' function for member m_mainwid_border
- int mainwid_spacing () const

  'Getter' function for member m_mainwid_spacing
- int control_height () const

  'Getter' function for member m_control_height
- int mainwid_x () const

  'Getter' function for member m_mainwid_x
- int mainwid_y () const

  'Getter' function for member m_mainwid_y
- void mainwnd_rows (int value)

  'Setter' function for member m_mainwnd_rows This value is not modified unless the value parameter is between 4 and 8, inclusive.
- void mainwnd_cols (int value)

  'Setter' function for member m_mainwnd_cols This value is not modified unless the value parameter is between 8 and 10, inclusive.
- void max_sets (int value)

  'Setter' function for member m_seqs_in_set
- void text_x (int value)

  'Setter' function for member m_max_sequence
- void text_y (int value)

  'Setter' function for member m_text_y This value is not modified unless the value parameter is between 12 and 12, inclusive.
- void seqchars_x (int value)

  'Setter' function for member m_seqchars_x This affects the size or crampiness of a pattern slot, and for now we will hardwire it to 15.
- void seqchars_y (int value)

  'Setter' function for member m_seqchars_y This affects the size or crampiness of a pattern slot, and for now we will hardwire it to 5.
- void seqarea_x (int value)

  'Setter' function for member m_seqarea_x
- void seqarea_y (int value)

  'Setter' function for member m_seqarea_y
- void seqarea_seq_x (int value)

  'Setter' function for member m_seqarea_seq_x
- void seqarea_seq_y (int value)

  'Setter' function for member m_seqarea_seq_y
- void mainwid_border (int value)

  'Setter' function for member m_mainwid_border This value is not modified unless the value parameter is between 0 and 3, inclusive.
- void mainwid_spacing (int value)

  'Setter' function for member m_mainwid_spacing This value is not modified unless the value parameter is between 2 and 6, inclusive.
- void control_height (int value)

  'Setter' function for member m_control_height This value is not modified unless the value parameter is between 0 and 4, inclusive.
- void dump_summary ()

  'Setter' function for member m_mainwid_y

7.30.1 Detailed Description

These settings will eventually be made part of the "user" settings file.

7.30.2 Member Function Documentation

7.30.2.1 void user_settings::set_defaults ( )

For the m_midi_buses and m_instruments members, this function can only iterate over the current size of the vectors. But the default size is zero!

7.30.2.2 void user_settings::set_globals ( ) const

Should be called at initialization, and after settings are read from the "user" configuration file.

7.30.2.3 void user_settings::get_globals ( )

Should be called before settings are written to the "user" configuration file.

7.30.2.4 const user_midi_bus& user_settings::bus (int *index*) `[inline]`

Cannot append the const specifier.

7.30.2.5 const user_instrument& user_settings::instrument (int *index*) `[inline]`

Cannot append the const specifier.

7.30.2.6 int user_settings::bus_instrument (int *buss*, int *channel*) `[inline]`

Todo Do this for controllers values and for user_instrument members.

7.30.2.7 void user_settings::mainwnd_rows (int *value*)

The default value is 4. Dependent values are recalculated after the assignment.

7.30.2.8 void user_settings::mainwnd_cols (int *value*)

The default value is 8. Dependent values are recalculated after the assignment.

7.30.2.9 void user_settings::max_sets (int *value*)

Warning

This is a dependent value at present, and changing it is experimental.

void user_settings::seqs_in_set (int value) { m_seqs_in_set = value; } *'Setter' function* for member *m_gmute_tracks*

Warning

> This is a dependent value at present, and changing it is experimental.

void user_settings::gmute_tracks (int value) { m_gmute_tracks = value; } *'Setter' function* for member *m_max_sets* This value is not modified unless the value parameter is between 32 and 64, inclusive. The default value is 32. Dependent values are recalculated after the assignment.

7.30.2.10 void user_settings::text_x (int *value*)

Warning

> This is a dependent value at present, and changing it is experimental.

void user_settings::max_sequence (int value) { m_max_sequence = value; } *'Setter' function* for member *m_text↩ _x* This value is not modified unless the value parameter is between 6 and 6, inclusive. The default value is 6. Dependent values are recalculated after the assignment. This value is currently restricted, until we can code up a bigger font.

7.30.2.11 void user_settings::text_y (int *value*)

The default value is 12. Dependent values are recalculated after the assignment. This value is currently restricted, until we can code up a bigger font.

7.30.2.12 void user_settings::mainwid_border (int *value*)

The default value is 0. Dependent values are recalculated after the assignment.

7.30.2.13 void user_settings::mainwid_spacing (int *value*)

The default value is 2. Dependent values are recalculated after the assignment.

7.30.2.14 void user_settings::control_height (int *value*)

The default value is 0. Dependent values are recalculated after the assignment.

7.30.2.15 void user_settings::dump_summary ( )

Warning

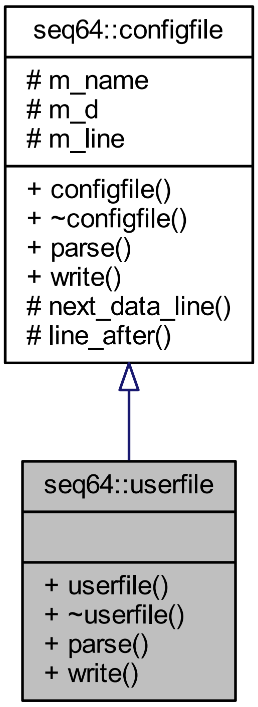
> This is a dependent value at present, and changing it is experimental.

void user_settings::mainwid_y (int value) { m_mainwid_y = value; } Provides a debug dump of basic information to help debug a surprisingly intractable problem with all busses having the name and values of the last buss in the configuration. Does its work only if PLATFORM_DEBUG and USE_DUMP_SUMMARY are defined. Only enabled in emergencies :-D.

7.31 seq64::userfile Class Reference

Supports the user's ∼/.seq24usr configuration file.

Inheritance diagram for seq64::userfile:



Public Member Functions

- userfile (const std::string &a_name)

  Principal constructor.
- ∼userfile ()

  A rote destructor needed for a derived class.
- bool parse (perform &a_perf)

  Parses a "usr" file, filling in the given perform object.
- bool write (const perform &a_perf)

  This function just returns false, as there is no "perform" information in the user-file yet.

Additional Inherited Members

7.31.1 Member Function Documentation

7.31.1.1 bool seq64::userfile::parse (perform & *a_perf*) [virtual]

This function opens the file as a text file (line-oriented).

Parameters

a_perf	The performance object, currently unused.

Implements seq64::configfile.

7.31.1.2 bool seq64::userfile::write (const perform & *a_perf*) [virtual]

Parameters

a_perf	The performance object, currently unused.

Implements seq64::configfile.

Index

∼keys_perform
    seq64::keys_perform, 29
∼perform
    seq64::perform, 47

add
    seq64::event_list, 25
add_event
    seq64::sequence, 64, 68
add_note
    seq64::sequence, 68
add_sequence
    seq64::perform, 47
add_trigger
    seq64::sequence, 64
all_notes_off
    seq64::perform, 50
append_sysex
    seq64::event, 23

bus
    user_settings, 79
bus_instrument
    user_settings, 79

change_event_data_range
    seq64::sequence, 69
channel_count
    user_midi_bus, 75
channel_max
    user_midi_bus, 75
clear_sequence_triggers
    seq64::perform, 47
clear_triggers
    seq64::sequence, 67
click
    seq64::click, 16
configfile
    seq64::configfile, 19
control_height
    user_settings, 80
controller_active
    user_instrument, 74
controller_max
    user_instrument, 73
controller_name
    user_instrument, 73
copy_selected
    seq64::sequence, 68
copy_triggers
    seq64::perform, 48
    seq64::sequence, 67
count
    seq64::event_list, 25

decrement_bpm
    seq64::perform, 52
decrement_selected
    seq64::sequence, 70
del_trigger
    seq64::sequence, 65
dump_summary
    user_settings, 80

e_deselect
    seq64::sequence, 60
e_remove_one
    seq64::sequence, 60
e_select
    seq64::sequence, 60
e_toggle_selection
    seq64::sequence, 60
event_count
    seq64::sequence, 61
event_list
    seq64::event_list, 24

fill
    seq64::midi_container, 35
fill_container
    seq64::sequence, 71

get
    seq64::midi_container, 35
    seq64::midi_list, 37
    seq64::midi_vector, 38
get_bw
    seq64::sequence, 61
get_data
    seq64::event, 23
get_global
    user_instrument, 73
    user_midi_bus, 75
get_globals
    user_settings, 79
get_keys
    seq64::keys_perform, 30
get_max_trigger
    seq64::perform, 52
    seq64::sequence, 66

get_midi_control_off
seq64::perform, 48
get_midi_control_on
seq64::perform, 48
get_midi_control_toggle
seq64::perform, 48
get_next_event
seq64::sequence, 71
get_next_note_event
seq64::sequence, 71
get_num_selected_events
seq64::sequence, 68
get_num_selected_notes
seq64::sequence, 68
get_rank
seq64::event, 23
get_screen_set_notepad
seq64::perform, 48
get_selected_trigger_end_tick
seq64::sequence, 66
get_selected_trigger_start_tick
seq64::sequence, 66
grow_selected
seq64::sequence, 70
grow_trigger
seq64::sequence, 65
gui_assistant
seq64::gui_assistant, 25

home_config_directory
rc_settings, 55

increment_bpm
seq64::perform, 52
increment_selected
seq64::sequence, 69
init
seq64::jack_assistant, 27
seq64::perform, 47
init_jack
seq64::perform, 47
instrument
user_midi_bus, 76
user_settings, 79
intersectEvents
seq64::sequence, 66
intersectNotes
seq64::sequence, 65
intersectTriggers
seq64::sequence, 65
is_active
seq64::perform, 50
is_dirty_edit
seq64::perform, 50
seq64::sequence, 63
is_dirty_main
seq64::perform, 50
seq64::sequence, 63
is_dirty_names
seq64::perform, 51
seq64::sequence, 63
is_dirty_perf
seq64::perform, 50
seq64::sequence, 63
is_letter
seq64::keystroke, 32
is_sequence_invalid
seq64::perform, 48
is_sequence_valid
seq64::perform, 48

jack_assistant
seq64::jack_assistant, 27
jack_shutdown
seq64::jack_assistant, 28
jack_sync_callback
seq64::jack_assistant, 28
jack_timebase_callback
seq64::jack_assistant, 28

key_name
seq64::keys_perform, 30
keystroke
seq64::keystroke, 31, 32

lash
seq64::lash, 33
launch_input_thread
seq64::perform, 47
launch_output_thread
seq64::perform, 47
line_after
seq64::configfile, 19
link_new
seq64::sequence, 70

m_line
seq64::configfile, 19
mainwid_border
user_settings, 80
mainwid_spacing
user_settings, 80
mainwnd_cols
user_settings, 79
mainwnd_key_event
seq64::perform, 53
mainwnd_rows
user_settings, 79
mark_selected
seq64::sequence, 70
max_sets
user_settings, 79
midifile
seq64::midifile, 38
mod_timestamp
seq64::event, 22
move_selected_triggers_to
seq64::sequence, 66

move_triggers
  seq64::perform, 48
  seq64::sequence, 66

new_sequence
  seq64::perform, 51
next_data_line
  seq64::configfile, 19

off_playing_notes
  seq64::sequence, 71
off_queued
  seq64::sequence, 63
operator<
  seq64::event, 22
operator=
  seq64::click, 16
  seq64::event_list, 24
  seq64::keystroke, 32
  seq64::sequence, 61
output
  seq64::jack_assistant, 27
output_func
  seq64::perform, 52

parse
  seq64::midifile, 39
  seq64::optionsfile, 41
  seq64::userfile, 81
paste_selected
  seq64::sequence, 68
perform
  seq64::perform, 46
perfroll_key_event
  seq64::perform, 53
play
  seq64::perform, 51
  seq64::sequence, 64
play_note_off
  seq64::sequence, 71
play_note_on
  seq64::sequence, 71
pop_redo
  seq64::sequence, 61
pop_undo
  seq64::sequence, 61
position
  seq64::jack_assistant, 27
  seq64::midi_container, 35
position_jack
  seq64::perform, 49
print
  seq64::sequence, 64
print_triggers
  seq64::sequence, 64
push_trigger_undo
  seq64::sequence, 61
push_undo
  seq64::sequence, 61
put
  seq64::midi_container, 35
  seq64::midi_list, 36
  seq64::midi_vector, 38

rc_settings, 53
  home_config_directory, 55
remove_marked
  seq64::sequence, 70
reset_draw_marker
  seq64::sequence, 71
reset_sequences
  seq64::perform, 51

select_action_e
  seq64::sequence, 60
select_all
  seq64::sequence, 68
select_events
  seq64::sequence, 67, 68
select_mute_group
  seq64::perform, 49
select_note_events
  seq64::sequence, 67
seq64::automutex, 15
seq64::click, 15
  click, 16
  operator=, 16
seq64::condition_var, 17
seq64::configfile, 18
  configfile, 19
  line_after, 19
  m_line, 19
  next_data_line, 19
seq64::event, 19
  append_sysex, 23
  get_data, 23
  get_rank, 23
  mod_timestamp, 22
  operator<, 22
  set_data, 22
  set_status, 22
seq64::event_list, 23
  add, 25
  count, 25
  event_list, 24
  operator=, 24
seq64::gui_assistant, 25
  gui_assistant, 25
seq64::gui_play_base, 26
seq64::jack_assistant, 26
  init, 27
  jack_assistant, 27
  jack_shutdown, 28
  jack_sync_callback, 28
  jack_timebase_callback, 28
  output, 27
  position, 27
seq64::jack_scratchpad, 28

seq64::keys_perform, 28
~keys_perform, 29
get_keys, 30
key_name, 30
set_all_key_events, 30
set_all_key_groups, 30
set_key_event, 30
set_key_group, 30
set_keys, 29
show_ui_sequence_key, 30
seq64::keys_perform_transfer, 31
seq64::keystroke, 31
is_letter, 32
keystroke, 31, 32
operator=, 32
seq64::lash, 32
lash, 33
set_alsa_client_id, 33
seq64::midi_container, 33
fill, 35
get, 35
position, 35
put, 35
seq64::midi_list, 35
get, 37
put, 36
seq64::midi_vector, 37
get, 38
put, 38
seq64::midifile, 38
midifile, 38
parse, 39
seq64::mutex, 39
seq64::optionsfile, 40
parse, 41
write, 42
seq64::perform, 42
~perform, 47
add_sequence, 47
all_notes_off, 50
clear_sequence_triggers, 47
copy_triggers, 48
decrement_bpm, 52
get_max_trigger, 52
get_midi_control_off, 48
get_midi_control_on, 48
get_midi_control_toggle, 48
get_screen_set_notepad, 48
increment_bpm, 52
init, 47
init_jack, 47
is_active, 50
is_dirty_edit, 50
is_dirty_main, 50
is_dirty_names, 51
is_dirty_perf, 50
is_sequence_invalid, 48
is_sequence_valid, 48
launch_input_thread, 47
launch_output_thread, 47
mainwnd_key_event, 53
move_triggers, 48
new_sequence, 51
output_func, 52
perform, 46
perfroll_key_event, 53
play, 51
position_jack, 49
reset_sequences, 51
select_mute_group, 49
set_bpm, 51
set_input_bus, 53
set_offset, 52
set_orig_ticks, 51
set_playing_screenset, 49
set_screen_set_notepad, 49
set_screenset, 49
set_sequence_control_status, 51
set_was_active, 50
show_ui_sequence_key, 52
start, 49
start_playing, 52
stop, 49
unset_mode_group_learn, 49
unset_sequence_control_status, 52
seq64::performcallback, 53
seq64::sequence, 55
add_event, 64, 68
add_note, 68
add_trigger, 64
change_event_data_range, 69
clear_triggers, 67
copy_selected, 68
copy_triggers, 67
decrement_selected, 70
del_trigger, 65
e_deselect, 60
e_remove_one, 60
e_select, 60
e_toggle_selection, 60
event_count, 61
fill_container, 71
get_bw, 61
get_max_trigger, 66
get_next_event, 71
get_next_note_event, 71
get_num_selected_events, 68
get_num_selected_notes, 68
get_selected_trigger_end_tick, 66
get_selected_trigger_start_tick, 66
grow_selected, 70
grow_trigger, 65
increment_selected, 69
intersectEvents, 66
intersectNotes, 65
intersectTriggers, 65

is_dirty_edit, 63
is_dirty_main, 63
is_dirty_names, 63
is_dirty_perf, 63
link_new, 70
mark_selected, 70
move_selected_triggers_to, 66
move_triggers, 66
off_playing_notes, 71
off_queued, 63
operator=, 61
paste_selected, 68
play, 64
play_note_off, 71
play_note_on, 71
pop_redo, 61
pop_undo, 61
print, 64
print_triggers, 64
push_trigger_undo, 61
push_undo, 61
remove_marked, 70
reset_draw_marker, 71
select_action_e, 60
select_all, 68
select_events, 67, 68
select_note_events, 67
set_bpm, 61
set_bw, 61
set_dirty, 64
set_dirty_mp, 63
set_length, 61
set_master_midi_bus, 67
set_midi_bus, 67
set_midi_channel, 64
set_orig_tick, 64
set_playing, 61
set_quantized_rec, 63
set_rec_vol, 61
set_recording, 63
set_snap_tick, 63
set_thru, 63
split_trigger, 65
stream_event, 69
stretch_selected, 70
toggle_queued, 63
transpose_notes, 72
unpaint_all, 70
unselect, 70
verify_and_link, 70
zero_markers, 70

seq64::trigger, 72
seq64::userfile, 80
parse, 81
write, 82
set_all_key_events
seq64::keys_perform, 30
set_all_key_groups
seq64::keys_perform, 30
set_alsa_client_id
seq64::lash, 33
set_bpm
seq64::perform, 51
seq64::sequence, 61
set_bw
seq64::sequence, 61
set_controller
user_instrument, 74
set_data
seq64::event, 22
set_defaults
user_instrument, 73
user_midi_bus, 75
user_settings, 79
set_dirty
seq64::sequence, 64
set_dirty_mp
seq64::sequence, 63
set_global
user_instrument, 73
user_midi_bus, 75
set_globals
user_settings, 79
set_input_bus
seq64::perform, 53
set_instrument
user_midi_bus, 76
set_key_event
seq64::keys_perform, 30
set_key_group
seq64::keys_perform, 30
set_keys
seq64::keys_perform, 29
set_length
seq64::sequence, 61
set_master_midi_bus
seq64::sequence, 67
set_midi_bus
seq64::sequence, 67
set_midi_channel
seq64::sequence, 64
set_offset
seq64::perform, 52
set_orig_tick
seq64::sequence, 64
set_orig_ticks
seq64::perform, 51
set_playing
seq64::sequence, 61
set_playing_screenset
seq64::perform, 49
set_quantized_rec
seq64::sequence, 63
set_rec_vol
seq64::sequence, 61
set_recording

seq64::sequence, 63
set_screen_set_notepad
seq64::perform, 49
set_screenset
seq64::perform, 49
set_sequence_control_status
seq64::perform, 51
set_snap_tick
seq64::sequence, 63
set_status
seq64::event, 22
set_thru
seq64::sequence, 63
set_was_active
seq64::perform, 50
show_ui_sequence_key
seq64::keys_perform, 30
seq64::perform, 52
split_trigger
seq64::sequence, 65
start
seq64::perform, 49
start_playing
seq64::perform, 52
stop
seq64::perform, 49
stream_event
seq64::sequence, 69
stretch_selected
seq64::sequence, 70

text_x
user_settings, 80
text_y
user_settings, 80
toggle_queued
seq64::sequence, 63
transpose_notes
seq64::sequence, 72

unpaint_all
seq64::sequence, 70
unselect
seq64::sequence, 70
unset_mode_group_learn
seq64::perform, 49
unset_sequence_control_status
seq64::perform, 52
user_instrument, 72
controller_active, 74
controller_max, 73
controller_name, 73
get_global, 73
set_controller, 74
set_defaults, 73
set_global, 73
user_instrument_t, 74
user_midi_bus, 74
channel_count, 75
channel_max, 75
get_global, 75
instrument, 76
set_defaults, 75
set_global, 75
set_instrument, 76
user_midi_bus_t, 76
user_settings, 76
bus, 79
bus_instrument, 79
control_height, 80
dump_summary, 80
get_globals, 79
instrument, 79
mainwid_border, 80
mainwid_spacing, 80
mainwnd_cols, 79
mainwnd_rows, 79
max_sets, 79
set_defaults, 79
set_globals, 79
text_x, 80
text_y, 80

verify_and_link
seq64::sequence, 70

write
seq64::optionsfile, 42
seq64::userfile, 82

zero_markers
seq64::sequence, 70