

Sequencer64 Developer/Tester's Reference Manual

0.9.9.5

Generated by Doxygen 1.8.9.1

Sun Oct 18 2015 14:59:09

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	Button Clicks on a Pattern	4
2.3.2	Patterns Window Key Shortcuts	5
2.3.3	The Sequencer64 User File	5
2.4	Tests Using Valgrind	5
2.4.1	Valgrind Suppressions	5
2.4.2	Full Valgrind Leak-Checking	6
2.4.2.1	Leak-Checking Basic Operation	6
2.5	Specific Fault Debugging	6
2.6	Snipping of a MIDI file.	6
3	Licenses	9
3.1	License Terms for the This Project.	9
3.2	XPC Application License	9
3.3	XPC Library License	10
3.4	XPC Documentation License	10
3.5	XPC Affero License	10
3.6	XPC License Summary	11
4	Todo List	13
5	Hierarchical Index	15
5.1	Class Hierarchy	15
6	Data Structure Index	17
6.1	Data Structures	17
7	Data Structure Documentation	21

7.1	seq64::AbstractPerfInput Class Reference	21
7.2	seq64::click Class Reference	21
7.2.1	Detailed Description	22
7.2.2	Constructor & Destructor Documentation	23
7.2.2.1	click	23
7.2.2.2	click	23
7.2.2.3	click	23
7.2.3	Member Function Documentation	23
7.2.3.1	operator=	23
7.2.4	Field Documentation	23
7.2.4.1	m_x	23
7.2.4.2	m_y	23
7.2.4.3	m_button	23
7.2.4.4	m_modifier	24
7.3	seq64::configfile Class Reference	24
7.3.1	Constructor & Destructor Documentation	25
7.3.1.1	configfile	25
7.3.2	Member Function Documentation	25
7.3.2.1	next_data_line	25
7.3.2.2	line_after	25
7.3.3	Field Documentation	25
7.3.3.1	m_line	25
7.4	seq64::event Class Reference	25
7.4.1	Detailed Description	28
7.4.2	Member Function Documentation	28
7.4.2.1	operator<	28
7.4.2.2	mod_timestamp	28
7.4.2.3	set_status	29
7.4.2.4	set_data	29
7.4.2.5	set_data	29
7.4.2.6	get_data	29
7.4.2.7	append_sysex	29
7.4.2.8	get_rank	30
7.4.3	Field Documentation	30
7.4.3.1	m_status	30
7.4.3.2	m_data	30
7.4.3.3	m_sysex	30
7.4.3.4	m_has_link	30
7.5	seq64::event_list::event_key Class Reference	30
7.5.1	Detailed Description	31

7.5.2	Constructor & Destructor Documentation	31
7.5.2.1	event_key	31
7.5.2.2	event_key	31
7.5.3	Member Function Documentation	31
7.5.3.1	operator<	31
7.5.4	Field Documentation	31
7.5.4.1	m_timestamp	31
7.5.4.2	m_rank	31
7.6	seq64::event_list Class Reference	31
7.6.1	Detailed Description	33
7.6.2	Constructor & Destructor Documentation	33
7.6.2.1	event_list	33
7.6.3	Member Function Documentation	33
7.6.3.1	operator=	33
7.6.3.2	count	33
7.6.3.3	add	34
7.6.3.4	merge	34
7.6.3.5	link_new	34
7.6.3.6	verify_and_link	34
7.6.3.7	mark_out_of_range	35
7.6.3.8	count_selected_events	35
7.7	seq64::font Class Reference	35
7.7.1	Member Enumeration Documentation	36
7.7.1.1	Color	36
7.7.2	Member Function Documentation	36
7.7.2.1	init	36
7.7.2.2	render_string_on_drawable	36
7.7.3	Field Documentation	36
7.7.3.1	m_pixmap	36
7.7.3.2	m_black_pixmap	37
7.7.3.3	m_white_pixmap	37
7.7.3.4	m_b_on_y_pixmap	37
7.7.3.5	m_y_on_b_pixmap	37
7.7.3.6	m_clip_mask	37
7.8	seq64::gui_assistant Class Reference	37
7.8.1	Detailed Description	38
7.8.2	Constructor & Destructor Documentation	39
7.8.2.1	gui_assistant	39
7.9	seq64::gui_assistant_gtk2 Class Reference	39
7.9.1	Field Documentation	40

7.9.1.1	sm_internal_keys	40
7.10	seq64::gui_drawingarea_gtk2 Class Reference	40
7.10.1	Detailed Description	41
7.10.2	Member Function Documentation	41
7.10.2.1	on_realize	41
7.10.3	Field Documentation	42
7.10.3.1	m_mainperf	42
7.10.3.2	m_window_x	42
7.10.3.3	m_drop_x	42
7.11	seq64::gui_palette_gtk2 Class Reference	42
7.11.1	Detailed Description	43
7.11.2	Constructor & Destructor Documentation	43
7.11.2.1	gui_palette_gtk2	43
7.12	seq64::gui_window_gtk2 Class Reference	43
7.12.1	Constructor & Destructor Documentation	45
7.12.1.1	gui_window_gtk2	45
7.12.2	Field Documentation	45
7.12.2.1	m_window_x	45
7.13	seq64::jack_assistant Class Reference	45
7.13.1	Constructor & Destructor Documentation	46
7.13.1.1	jack_assistant	46
7.13.2	Member Function Documentation	46
7.13.2.1	init	46
7.13.2.2	position	46
7.13.2.3	output	46
7.13.3	Friends And Related Function Documentation	47
7.13.3.1	jack_sync_callback	47
7.13.3.2	jack_shutdown	47
7.13.3.3	jack_timebase_callback	47
7.14	seq64::jack_scratchpad Struct Reference	47
7.14.1	Detailed Description	47
7.15	seq64::keybindentry Class Reference	47
7.15.1	Member Enumeration Documentation	48
7.15.1.1	type	48
7.15.2	Constructor & Destructor Documentation	48
7.15.2.1	keybindentry	48
7.15.3	Member Function Documentation	49
7.15.3.1	set	49
7.15.3.2	on_key_press_event	49
7.15.4	Field Documentation	49

7.15.4.1	<code>m_key</code>	49
7.16	<code>seq64::keys_perform</code> Class Reference	49
7.16.1	Detailed Description	51
7.16.2	Constructor & Destructor Documentation	51
7.16.2.1	<code>~keys_perform</code>	51
7.16.3	Member Function Documentation	51
7.16.3.1	<code>set_keys</code>	51
7.16.3.2	<code>get_keys</code>	52
7.16.3.3	<code>show_ui_sequence_key</code>	52
7.16.3.4	<code>key_name</code>	52
7.16.3.5	<code>set_all_key_events</code>	52
7.16.3.6	<code>set_all_key_groups</code>	52
7.16.3.7	<code>set_key_event</code>	52
7.16.3.8	<code>set_key_group</code>	52
7.16.4	Field Documentation	53
7.16.4.1	<code>m_key_bpm_up</code>	53
7.17	<code>seq64::keys_perform_gtk2</code> Class Reference	53
7.17.1	Detailed Description	55
7.17.2	Constructor & Destructor Documentation	55
7.17.2.1	<code>~keys_perform_gtk2</code>	55
7.17.3	Member Function Documentation	55
7.17.3.1	<code>key_name</code>	55
7.17.3.2	<code>set_all_key_events</code>	55
7.17.3.3	<code>set_all_key_groups</code>	55
7.18	<code>seq64::keys_perform_transfer</code> Struct Reference	55
7.19	<code>seq64::keystroke</code> Class Reference	55
7.19.1	Detailed Description	56
7.19.2	Constructor & Destructor Documentation	56
7.19.2.1	<code>keystroke</code>	56
7.19.2.2	<code>keystroke</code>	57
7.19.3	Member Function Documentation	57
7.19.3.1	<code>operator=</code>	57
7.19.3.2	<code>is_letter</code>	57
7.19.4	Field Documentation	57
7.19.4.1	<code>m_is_press</code>	57
7.19.4.2	<code>m_key</code>	57
7.19.4.3	<code>m_modifier</code>	58
7.20	<code>seq64::lash</code> Class Reference	58
7.20.1	Detailed Description	58
7.20.2	Constructor & Destructor Documentation	58

7.20.2.1	lash	58
7.20.3	Member Function Documentation	59
7.20.3.1	set_alsa_client_id	59
7.20.3.2	process_events	59
7.20.3.3	init	59
7.20.3.4	handle_event	59
7.20.3.5	handle_config	59
7.21	seq64::maintime Class Reference	59
7.21.1	Constructor & Destructor Documentation	61
7.21.1.1	maintime	61
7.21.2	Member Function Documentation	61
7.21.2.1	idle_progress	61
7.21.2.2	on_realize	61
7.22	seq64::mainwid Class Reference	61
7.22.1	Constructor & Destructor Documentation	64
7.22.1.1	mainwid	64
7.22.2	Member Function Documentation	64
7.22.2.1	set_screenset	64
7.22.2.2	update_sequence_on_window	64
7.22.2.3	update_markers	64
7.22.2.4	draw_marker_on_sequence	64
7.22.2.5	valid_sequence	64
7.22.2.6	draw_sequence_on_pixmap	65
7.22.2.7	draw_sequences_on_pixmap	65
7.22.2.8	draw_sequence_pixmap_on_window	65
7.22.2.9	seq_from_xy	65
7.22.2.10	timeout	66
7.22.2.11	redraw	66
7.22.2.12	calculate_base_sizes	66
7.22.2.13	on_realize	66
7.22.2.14	on_expose_event	66
7.22.2.15	on_button_press_event	66
7.22.2.16	on_button_release_event	67
7.22.2.17	on_motion_notify_event	67
7.22.2.18	on_focus_in_event	67
7.22.2.19	on_focus_out_event	67
7.23	seq64::mainwnd Class Reference	67
7.23.1	Constructor & Destructor Documentation	71
7.23.1.1	mainwnd	71
7.23.2	Member Function Documentation	71

7.23.2.1	open_file	71
7.23.2.2	ppqn	71
7.23.2.3	file_import_dialog	72
7.23.2.4	about_dialog	72
7.23.2.5	adj_callback_ss	72
7.23.2.6	timer_callback	72
7.23.2.7	open_performance_edit	72
7.23.2.8	update_window_title	72
7.23.2.9	save_file	72
7.23.2.10	signal_action	73
7.23.2.11	on_delete_event	73
7.23.2.12	on_key_press_event	73
7.23.2.13	on_key_release_event	73
7.23.2.14	on_grouplearnchange	73
7.23.3	Field Documentation	73
7.23.3.1	m_sigpipe	73
7.23.3.2	m_main_wid	73
7.23.3.3	m_spinbutton_load_offset	73
7.24	seq64::midi_container Class Reference	74
7.24.1	Member Function Documentation	75
7.24.1.1	fill	75
7.24.1.2	put	75
7.24.1.3	get	75
7.24.1.4	position	76
7.24.1.5	add_variable	76
7.24.1.6	add_long	76
7.25	seq64::midi_list Class Reference	76
7.25.1	Member Typedef Documentation	78
7.25.1.1	CharList	78
7.25.2	Member Function Documentation	78
7.25.2.1	put	78
7.25.2.2	get	78
7.26	seq64::midi_vector Class Reference	78
7.26.1	Member Function Documentation	80
7.26.1.1	put	80
7.26.1.2	get	80
7.27	seq64::midifile Class Reference	80
7.27.1	Detailed Description	82
7.27.2	Constructor & Destructor Documentation	82
7.27.2.1	midifile	82

7.27.3	Member Function Documentation	82
7.27.3.1	parse	82
7.27.3.2	write	83
7.27.3.3	ppqn	83
7.27.3.4	parse_prop_header	83
7.27.3.5	parse_proprietary_track	84
7.27.3.6	read_long	84
7.27.3.7	read_short	84
7.27.3.8	read_varinum	84
7.27.3.9	write_long	85
7.27.3.10	write_short	85
7.27.3.11	write_byte	85
7.27.3.12	write_varinum	85
7.27.3.13	write_track_name	85
7.27.3.14	write_seq_number	85
7.27.3.15	write_prop_header	86
7.27.3.16	write_proprietary_track	86
7.27.3.17	varinum_size	87
7.27.3.18	prop_item_size	87
7.27.3.19	seq_number_size	87
7.27.4	Field Documentation	87
7.27.4.1	m_pos	87
7.27.4.2	m_data	87
7.27.4.3	m_char_list	87
7.27.4.4	m_new_format	87
7.28	seq64::options Class Reference	87
7.28.1	Field Documentation	88
7.28.1.1	m_notebook	88
7.29	seq64::optionsfile Class Reference	88
7.29.1	Detailed Description	89
7.29.2	Member Function Documentation	89
7.29.2.1	parse	89
7.29.2.2	write	90
7.30	seq64::perfedit Class Reference	90
7.30.1	Detailed Description	93
7.30.2	Constructor & Destructor Documentation	93
7.30.2.1	perfedit	93
7.30.2.2	~perfedit	93
7.30.3	Member Function Documentation	93
7.30.3.1	init_before_show	93

7.30.3.2	set_guides	93
7.30.3.3	expand	93
7.30.3.4	collapse	93
7.30.3.5	copy	93
7.30.3.6	undo	94
7.30.3.7	timeout	94
7.30.3.8	start_playing	94
7.31	seq64::perfnames Class Reference	94
7.31.1	Detailed Description	96
7.31.2	Constructor & Destructor Documentation	96
7.31.2.1	perfnames	96
7.31.3	Member Function Documentation	96
7.31.3.1	on_realize	96
7.31.3.2	on_expose_event	96
7.31.3.3	on_size_allocate	97
7.32	seq64::perform Class Reference	97
7.32.1	Detailed Description	102
7.32.2	Constructor & Destructor Documentation	102
7.32.2.1	perform	102
7.32.2.2	~perform	102
7.32.3	Member Function Documentation	102
7.32.3.1	sequence_count	102
7.32.3.2	init	102
7.32.3.3	clear_all	102
7.32.3.4	launch_input_thread	102
7.32.3.5	launch_output_thread	103
7.32.3.6	init_jack	103
7.32.3.7	add_sequence	103
7.32.3.8	delete_sequence	103
7.32.3.9	clear_sequence_triggers	103
7.32.3.10	move_triggers	103
7.32.3.11	copy_triggers	103
7.32.3.12	get_midi_control_toggle	103
7.32.3.13	get_midi_control_on	104
7.32.3.14	get_midi_control_off	104
7.32.3.15	get_screen_set_notepad	104
7.32.3.16	set_screen_set_notepad	104
7.32.3.17	set_screenset	104
7.32.3.18	set_playing_screenset	104
7.32.3.19	unset_mode_group_learn	105

7.32.3.20 select_mute_group	105
7.32.3.21 start	105
7.32.3.22 stop	105
7.32.3.23 position_jack	105
7.32.3.24 all_notes_off	105
7.32.3.25 set_active	105
7.32.3.26 set_was_active	105
7.32.3.27 is_active	105
7.32.3.28 is_dirty_main	106
7.32.3.29 is_dirty_edit	106
7.32.3.30 is_dirty_perf	106
7.32.3.31 is_dirty_names	106
7.32.3.32 new_sequence	106
7.32.3.33 get_sequence	107
7.32.3.34 reset_sequences	107
7.32.3.35 play	107
7.32.3.36 set_orig_ticks	107
7.32.3.37 set_bpm	107
7.32.3.38 set_sequence_control_status	107
7.32.3.39 unset_sequence_control_status	107
7.32.3.40 output_func	107
7.32.3.41 get_max_trigger	108
7.32.3.42 set_offset	108
7.32.3.43 show_ui_sequence_key	108
7.32.3.44 start_playing	108
7.32.3.45 decrement_bpm	108
7.32.3.46 increment_bpm	108
7.32.3.47 set_input_bus	108
7.32.3.48 mainwnd_key_event	108
7.32.3.49 perfroll_key_event	109
7.32.3.50 is_midi_control_valid	109
7.32.3.51 is_screenset_valid	109
7.32.3.52 is_seq_valid	109
7.32.3.53 is_mseq_valid	109
7.32.3.54 install_sequence	109
7.32.3.55 inner_start	110
7.32.3.56 set_key_event	110
7.32.3.57 set_key_group	110
7.32.3.58 clamp_track	110
7.32.4 Friends And Related Function Documentation	110

7.32.4.1	jack_sync_callback	110
7.32.5	Field Documentation	110
7.32.5.1	m_playback_mode	110
7.33	seq64::performcallback Struct Reference	110
7.33.1	Detailed Description	111
7.34	seq64::perfolr Class Reference	111
7.34.1	Member Function Documentation	114
7.34.1.1	update_sizes	114
7.34.1.2	init_before_show	114
7.34.1.3	convert_xy	114
7.34.1.4	convert_x	114
7.34.1.5	snap_x	115
7.34.1.6	start_playing	115
7.34.1.7	stop_playing	115
7.34.1.8	draw_sequence_on	115
7.34.1.9	on_realize	115
7.34.1.10	on_button_press_event	115
7.34.1.11	on_button_release_event	115
7.34.1.12	on_key_press_event	115
7.35	seq64::perftime Class Reference	115
7.35.1	Constructor & Destructor Documentation	117
7.35.1.1	perftime	117
7.35.2	Member Function Documentation	117
7.35.2.1	on_realize	117
7.35.2.2	on_expose_event	117
7.36	rc_settings Class Reference	118
7.36.1	Member Function Documentation	120
7.36.1.1	home_config_directory	120
7.36.1.2	make_directory	120
7.37	seq64::gui_drawingarea_gtk2::rect Struct Reference	120
7.38	seq64::rect Class Reference	120
7.39	seq64::Seq24PerfInput Class Reference	120
7.39.1	Member Function Documentation	121
7.39.1.1	on_button_press_event	121
7.39.1.2	on_button_release_event	122
7.39.1.3	set_adding	122
7.40	seq64::Seq24SeqEventInput Struct Reference	122
7.40.1	Member Function Documentation	122
7.40.1.1	set_adding	122
7.40.1.2	on_button_press_event	122

7.41	seq64::Seq24SeqRollInput Struct Reference	122
7.41.1	Member Function Documentation	123
7.41.1.1	set_adding	123
7.42	seq64::seqdata Class Reference	123
7.42.1	Constructor & Destructor Documentation	126
7.42.1.1	seqdata	126
7.42.2	Member Function Documentation	126
7.42.2.1	reset	126
7.42.2.2	redraw	126
7.42.2.3	set_zoom	126
7.42.2.4	idle_redraw	126
7.42.2.5	update_sizes	126
7.42.2.6	xy_to_rect	126
7.42.2.7	on_realize	126
7.42.2.8	on_motion_notify_event	127
7.42.2.9	on_scroll_event	127
7.43	seq64::seqedit Class Reference	127
7.43.1	Detailed Description	131
7.43.2	Constructor & Destructor Documentation	131
7.43.2.1	seqedit	131
7.43.3	Member Function Documentation	131
7.43.3.1	set_zoom	131
7.43.3.2	set_snap	131
7.43.3.3	set_note_length	131
7.43.3.4	apply_length	131
7.43.3.5	get_measures	131
7.43.3.6	set_scale	132
7.43.3.7	set_key	132
7.43.3.8	set_background_sequence	132
7.43.3.9	name_change_callback	132
7.43.3.10	set_data_type	132
7.43.3.11	popup_event_menu	132
7.43.3.12	popup_midibus_menu	132
7.43.3.13	popup_sequence_menu	132
7.43.3.14	popup_tool_menu	132
7.43.3.15	do_action	132
7.43.3.16	on_delete_event	132
7.43.4	Field Documentation	133
7.43.4.1	mc_min_zoom	133
7.44	seq64::seqevent Class Reference	133

7.44.1	Member Function Documentation	136
7.44.1.1	set_snap	136
7.44.1.2	set_data_type	136
7.44.1.3	update_sizes	136
7.44.1.4	draw_background	136
7.44.1.5	draw_pixmap_on_window	136
7.44.1.6	idle_redraw	137
7.44.1.7	x_to_w	137
7.44.1.8	drop_event	137
7.44.1.9	start_paste	137
7.44.1.10	convert_x	137
7.44.1.11	convert_t	137
7.44.1.12	snap_x	137
7.44.1.13	on_realize	137
7.44.1.14	on_button_press_event	137
7.44.1.15	on_button_release_event	138
7.44.1.16	on_motion_notify_event	138
7.44.1.17	on_key_press_event	138
7.45	seq64::seqkeys Class Reference	138
7.45.1	Member Function Documentation	140
7.45.1.1	set_hint_state	140
7.45.1.2	draw_key	141
7.45.1.3	on_realize	141
7.45.1.4	on_button_press_event	141
7.45.1.5	on_button_release_event	141
7.46	seq64::seqmenu Class Reference	141
7.46.1	Detailed Description	143
7.46.2	Constructor & Destructor Documentation	143
7.46.2.1	seqmenu	143
7.46.2.2	~seqmenu	144
7.46.3	Member Function Documentation	144
7.46.3.1	seq_edit	144
7.46.3.2	seq_copy	144
7.46.3.3	seq_cut	144
7.46.3.4	seq_paste	144
7.46.3.5	seq_clear_perf	144
7.46.4	Field Documentation	144
7.46.4.1	m_seqedit	144
7.47	seq64::seqroll Class Reference	144
7.47.1	Member Function Documentation	147

7.47.1.1	reset	147
7.47.1.2	set_data_type	148
7.47.1.3	set_background_sequence	148
7.47.1.4	draw_events_on_pixmap	148
7.47.1.5	convert_tn	148
7.47.1.6	snap_x	148
7.47.1.7	on_key_press_event	148
7.47.1.8	on_scroll_event	148
7.48	seq64::seqtime Class Reference	148
7.48.1	Member Function Documentation	150
7.48.1.1	on_button_press_event	150
7.48.1.2	on_button_release_event	150
7.49	seq64::sequence Class Reference	150
7.49.1	Detailed Description	156
7.49.2	Member Enumeration Documentation	156
7.49.2.1	select_action_e	156
7.49.3	Member Function Documentation	156
7.49.3.1	operator=	156
7.49.3.2	event_count	156
7.49.3.3	push_undo	156
7.49.3.4	pop_undo	156
7.49.3.5	pop_redo	156
7.49.3.6	push_trigger_undo	157
7.49.3.7	set_bpm	157
7.49.3.8	set_bw	157
7.49.3.9	get_bw	157
7.49.3.10	set_rec_vol	157
7.49.3.11	set_length	157
7.49.3.12	set_playing	157
7.49.3.13	toggle_queued	157
7.49.3.14	off_queued	157
7.49.3.15	set_recording	157
7.49.3.16	set_snap_tick	157
7.49.3.17	set_quantized_rec	158
7.49.3.18	set_thru	158
7.49.3.19	is_dirty_main	158
7.49.3.20	is_dirty_edit	158
7.49.3.21	is_dirty_perf	158
7.49.3.22	is_dirty_names	158
7.49.3.23	set_dirty_mp	158

7.49.3.24 set_dirty	158
7.49.3.25 set_midi_channel	158
7.49.3.26 print	158
7.49.3.27 print_triggers	158
7.49.3.28 play	159
7.49.3.29 set_orig_tick	159
7.49.3.30 add_event	159
7.49.3.31 add_trigger	159
7.49.3.32 split_trigger	159
7.49.3.33 grow_trigger	160
7.49.3.34 del_trigger	160
7.49.3.35 intersectTriggers	160
7.49.3.36 intersectNotes	160
7.49.3.37 intersectEvents	160
7.49.3.38 move_selected_triggers_to	161
7.49.3.39 selected_trigger_start	161
7.49.3.40 selected_trigger_end	161
7.49.3.41 get_max_trigger	161
7.49.3.42 move_triggers	161
7.49.3.43 copy_triggers	161
7.49.3.44 clear_triggers	162
7.49.3.45 set_midi_bus	162
7.49.3.46 set_master_midi_bus	162
7.49.3.47 select_note_events	162
7.49.3.48 select_events	162
7.49.3.49 select_events	162
7.49.3.50 get_num_selected_notes	162
7.49.3.51 get_num_selected_events	163
7.49.3.52 select_all	163
7.49.3.53 copy_selected	163
7.49.3.54 paste_selected	163
7.49.3.55 add_note	163
7.49.3.56 add_event	163
7.49.3.57 stream_event	163
7.49.3.58 change_event_data_range	163
7.49.3.59 increment_selected	164
7.49.3.60 decrement_selected	164
7.49.3.61 grow_selected	164
7.49.3.62 stretch_selected	164
7.49.3.63 remove_marked	165

7.49.3.64	mark_selected	165
7.49.3.65	unpaint_all	165
7.49.3.66	unselect	165
7.49.3.67	verify_and_link	165
7.49.3.68	link_new	165
7.49.3.69	zero_markers	165
7.49.3.70	play_note_on	165
7.49.3.71	play_note_off	165
7.49.3.72	off_playing_notes	165
7.49.3.73	reset_draw_marker	165
7.49.3.74	get_next_note_event	166
7.49.3.75	get_next_event	166
7.49.3.76	get_next_event	166
7.49.3.77	fill_container	166
7.49.3.78	transpose_notes	166
7.49.3.79	put_event_on_bus	166
7.49.3.80	set_trigger_offset	166
7.49.3.81	split_trigger	166
7.49.3.82	adjust_trigger_offsets_to_length	167
7.49.3.83	remove	167
7.49.3.84	remove	167
7.49.4	Field Documentation	167
7.49.4.1	m_mutex	167
7.50	seq64::trigger Class Reference	167
7.50.1	Detailed Description	168
7.51	user_instrument Class Reference	168
7.51.1	Detailed Description	169
7.51.2	Member Function Documentation	169
7.51.2.1	set_defaults	169
7.51.2.2	set_global	169
7.51.2.3	get_global	169
7.51.2.4	controller_max	170
7.51.2.5	controller_name	170
7.51.2.6	controller_active	170
7.51.2.7	set_controller	170
7.51.2.8	set_name	170
7.51.2.9	copy_definitions	170
7.51.3	Field Documentation	170
7.51.3.1	m_is_valid	170
7.51.3.2	m_controller_count	171

7.52	user_instrument_t Struct Reference	171
7.53	user_midi_bus Class Reference	171
7.53.1	Detailed Description	172
7.53.2	Member Function Documentation	172
7.53.2.1	set_defaults	172
7.53.2.2	set_global	172
7.53.2.3	get_global	172
7.53.2.4	channel_count	172
7.53.2.5	channel_max	172
7.53.2.6	instrument	173
7.53.2.7	set_instrument	173
7.53.2.8	copy_definitions	173
7.53.3	Field Documentation	173
7.53.3.1	m_is_valid	173
7.53.3.2	m_channel_count	173
7.54	user_midi_bus_t Struct Reference	173
7.55	user_settings Class Reference	173
7.55.1	Detailed Description	177
7.55.2	Member Typedef Documentation	177
7.55.2.1	Busses	177
7.55.3	Member Function Documentation	177
7.55.3.1	set_defaults	177
7.55.3.2	set_globals	177
7.55.3.3	get_globals	177
7.55.3.4	bus	177
7.55.3.5	instrument	177
7.55.3.6	bus_instrument	177
7.55.3.7	mainwnd_rows	178
7.55.3.8	mainwnd_cols	178
7.55.3.9	max_sets	178
7.55.3.10	text_x	178
7.55.3.11	text_y	178
7.55.3.12	mainwid_border	178
7.55.3.13	mainwid_spacing	178
7.55.3.14	control_height	178
7.55.3.15	dump_summary	179
7.55.3.16	private_bus	179
7.55.3.17	private_instrument	179
7.55.4	Field Documentation	179
7.55.4.1	m_midi_buses	179

7.55.4.2	m_instruments	179
7.55.4.3	m_mainwnd_rows	179
7.55.4.4	m_mainwnd_cols	179
7.55.4.5	m_seqs_in_set	179
7.55.4.6	m_max_sets	179
7.55.4.7	m_text_x	180
7.55.4.8	m_seqchars_x	180
7.55.4.9	m_seqarea_x	180
7.55.4.10	m_seqarea_seq_x	180
7.55.4.11	m_mainwid_border	180
7.55.4.12	m_control_height	180
7.55.4.13	m_mainwid_x	180
7.56	seq64::userfile Class Reference	180
7.56.1	Member Function Documentation	181
7.56.1.1	parse	181
7.56.1.2	write	182
Index		183

Chapter 1

Sequencer64

Author(s) Chris Ahlstrom 2015-10-17

1.1 Introduction

Sequencer64 is a major 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-code 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 2015-10-18

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. If desired, use the `--bus` option described below to force the buss number to the buss you need, as shown in the second version of the command:

```
$ sequencer64/sequencer64 &
$ sequencer64/sequencer64 --bus 5 &
```

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 in Yoshimi?), 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".

The next instruction still works, but it is easier to simply pass the option `--bus 5` to Sequencer64 when starting it up.

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

The Patterns window is the inside portion of the main window, supported by the `mainwid` class. it contains a grid of boxes or slots, with each slot potentially containing a pattern, sequence, or track. Empty tracks (i.e. tracks that contain no events, like title-only tracks) are highlighted in yellow.

This window supports only a single variant of mouse-handling.

2.3.1 Button Clicks on a Pattern

A left-click on a pattern slot should cause the following to happen:

1. The pattern will be highlighted (white on a black background). This won't occur until the button is released.
2. During playback, the pattern will emit MIDI events and play its sequence.
3. If the pattern is dragged to another slot, whether playing is in progress or not, releasing the button in the destination slot will move the pattern to that slot.

A right-click on a pattern slot should cause the following to happen:

1. If the pattern is empty, then a pop-up menu to make a New pattern, paste a pattern, or make other selections will appear.
2. If the pattern is active, then a pop-up menu to Edit the pattern or make other selections will appear.
3. A second right-click, just off the menu, will dismiss the menu.

2.3.2 Patterns Window Key Shortcuts

First, note the selection of the File / Options / Keyboard / Show keys option. The tests here should work whether or not it is selected. The only difference is if the keys are shown.

We got a segfault during this test, when we weren't being systematic about it.

2.3.3 The Sequencer64 User File

To be discussed.

2.4 Tests Using Valgrind

Valgrind is a very useful tool for unearthing memory issues and other issues in an application, especially when one has the source code and can build the code with debugging information.

One runs the application from the command line, preceding its command line with valgrind and some of its options.

2.4.1 Valgrind Suppressions

One problem with valgrind is that it also uncovers errors in system libraries that one has no control over. These errors clutter the output, so we suppress them using a valgrind "suppressions" file. Here's how to create one:

```
$ valgrind --gen-suppressions=yes --log-file=val.supp ./Sequencer64/sequencer64
$ valgrind --gen-suppressions=all --log-file=val.supp ./Sequencer64/sequencer64
```

As the program runs, one is asked to print a suppression. If the error is due to a system or third-party library, answer "Y return", and then copy-and-paste the suppression to a file, giving it a name. For example, we provide a file `contrib/seq64.supp` containing suppressions of errors that annoy us. There are way too many "errors" in ALSA, GTK+, gtkmm, glibc, and more.

The second command collects all the suppressions. Passing the val.supp file through sed makes it immediately usable:

```
$ sed -i -e /^==/g val.supp
```

Running valgrind like this then shows mostly the errors we care about:

```
$ valgrind --suppressions=val.supp ./Sequencer64/sequencer64
```

We've added some other suppression files to the `contrib` directory. Too much! For example:

<https://github.com/dtrebbien/GNOME.supp>

However, overall this process is very painful, and we're going to eventually do all the valgrind work on the unit-test project for Sequencer64:

<https://github.com/ahlstromcj/seq64-tests>

2.4.2 Full Valgrind Leak-Checking

Here's how to capture errors, while suppressing the system errors and while generating a log file:

```
$ valgrind --suppressions=contrib/seq64.supp --leak-check=full \
  --track-origins=yes --log-file=valgrind.log --show-leak-kinds=all \
  ./Sequencer64/sequencer64
```

The errors can be also be re-routed to a log-file via the "2> valgrind.log" shell redirection.

Another idea is to precede the valgrind command with the following construct:

```
$ G_SLICE=debug-blocks valgrind ...
```

G_SLICE=debug-blocks will turn off gtk's advanced memory management to allow valgrind to show correct results. This results in an amazing plethora of invalid read and invalid write errors in GNOME-related libraries. Sheesh!

And don't forget about Valgrind's "massif" memory-tracking tool! (More to come!)

2.4.2.1 Leak-Checking Basic Operation

For the first pass, just run Sequencer64, then immediately exit. Then scan the log file to see if any "errors" can be pinpointed to the application and library code.

Don't forget to run the same scenario without valgrind, in a console window, to see if any of our own debug/problem output occurs.

In any case, leakage tagged as "still reachable" isn't as bad as leakage tagged as "definitely lost" or "indirectly lost".

But good luck finding a Sequencer64 bug buried in the chaff of 3rd-party valgrind reports, even with some suppressions enabled. Apparently a lot of them have to do with data structures that are intended to last the full life of the application.

One can make the search a little easier by searching for the "seq64" namespace in the valgrind log.

2.5 Specific Fault Debugging

This section goes through specific debugging cases we encountered. They should be part of the regular testing of Sequencer64.

2.6 Snipping of a MIDI file.

In order to have a test file for the *seq64-tests* project, we loaded up the *b4uacuse-GM-format.midi* file, removed all but four of the tracks, and saved it as *b4uacuse-snipped.midi*. Loading this file into Sequencer64 caused the following:

```
$ ./Sequencer64/sequencer64
[Reading user configuration /home/ahlstrom/.config/sequencer64/sequencer64.usr]
[Reading rc configuration /home/ahlstrom/.config/sequencer64/sequencer64.rc]
get_sequence(): m_seqs[4] not null
Segmentation fault
```

First step, fire up a debugger and see what happened. We use *cgdb*, a text-based front-end for gdb with a "vi" feel.

```
$ cgdb ./Sequencer64/sequencer64
```

Just hit "r", do *File / Open*, navigate to `b4uacuse-snipped.midi`, select it, and watch what happens.

The "bt" (backtrace) command shows a pretty large stack, 52 items. Page up to the top of the stack, and select frame 1 ("fr 1"). This shows a mutex at a very low address, 0x650! Frame 2 shows we are in the automutex constructor, calling `lock()` on that same badly-located mutex. Frame 3 is in `sequence::event_count()`, same bad mutex, and the `m_events` member is at address 0x0. Obviously, we're dealing with an unallocated sequence.

Frame 4 is in `mainwid::draw_sequence_on_pixmap()`, just after we've retrieved the next sequence via `perform::get_sequence(4)`. But that would be the fifth sequence (the sequence numbers start at 0), and we snipped all but 4 from the file before we saved it.

So, one thing we need to do is *check* the value returned by `get_sequence()` before we try to use it. The other thing to do is figure out how we got to the fifth sequence, and fix that code as well. Using the command "`p perf().sequence_count()`", we verify that there are indeed only 4 sequences allocated.

Frame 5 is in `mainwid::draw_sequences_on_pixmap()`. That function tries to load all sequences on the current screen-set, from 0 to 31, without checking to see how many there actually are. Inefficient and dangerous.

Frame 6 is in `mainwid::reset()`. We could pass `perf().sequence_count()` here for checking, or get it in `mainwid::draw_sequences_on_pixmap()`.

Before we fix this issue, we need to load a file that works, to see why it does not fail for most files. We will put a breakpoint at the top `mainwid::draw_sequences_on_pixmap()`.

We hit the breakpoint before even loading a file, with a `sequence_count()` of 0. The call to `valid_sequence(0)` passes the test. We may want to make `valid_sequence()` take the `sequence_count()` into account. But the call to `perf().is_active(0)` prevents anything bad from happening at startup time.

Once we load a good file, the `sequence_count()` is 14 in `mainwid::draw_sequences_on_pixmap()`. We turn on the display of "offset" using the command "display offset", and "c" (for "continue") until `offset = 14`, which means we are beyond that last sequence. That bad access is prevented by `perf().is_active(14)`.

So the fundamental problem is that `perf().is_active(4)` is not protecting the access when we load the "bad file". We need to find and fix that issue before papering over the problem with better access checks.

Start again, putting a breakpoint in the call to "new sequence(m_ppqn)" in `midifile`. This call sets up some members and clears the list of 256 playing notes. Add another breakpoint at "`a_perf.add_sequence()`" to see what's happening there.

What we find is that the first two tracks have proper sequence numbers as read from the MIDI file, 0 and 1. But the third one preserves the number from the old file, 4. We have a disjunction between the track number and the sequence number, a conceptual problem. We can leave it as is, and beef up the error-checking, or replace the sequence number with the track number when loading the file. What to do?

- Make sure that the is-active flag for all sequences is "false", that the pointers are always null, and make sure to test both of these items (depending on context) before doing anything with the sequence.
- Convert the sequence number to the track number upon saving the MIDI file, or upon reading the MIDI file, and use that number when adding the sequence to the perform object. This might affect some seq24/sequencer64 functionality, however. It's big move.

We need information on reading and importing.

First, if we look at a file that we created long ago by importing `b4uacuse.mid`, `b4uacuse-GM-format.midi`, it has its fourteen sequence numbers identical to their track numbers. No problem.

Second, if we just read `b4uacuse.mid`, a non-seq24-created MIDI file, we see that each of its tracks have no sequence number – they are all zero. The `perform::add_sequence()` simply iterates from the beginning of `m_seqs[]` until it finds an inactive `m_seqs[i]`, and uses that element to hold the sequence pointer.

But now it also segfaults! Let's fix all the non-checked `get_sequence()` calls right away, it is too big an issue to ignore.

In the end, we have to be aware that a screen-set can have blank (null) slots interspersed amongst the active slots.

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.

Copyright (C) 2015-2015 by Chris Ahlstrom

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

This program 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 Public License for more details.

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

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

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

<http://www.gnu.org/licenses/gpl-3.0.txt>

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 perroll module that we need to move here.

File `mainwnd.cpp`

Figure out best way to select non-legacy PPQN behavior, probably, for now, a command-line option.

- Add a GUI element that shows the actual PPQN in force, maybe next to the maintime object, or in the title caption.

Global `seq64::mainwnd::timeout ()`

We should use this callback to display the current time in the playback.

Global `seq64::mainwnd::file_import_dialog ()`

We need to look into the Import process and document it better.

Global `seq64::mainwnd::mainwnd (perform &a_p)`

Offload most of the work into an initialization function like options does; make the perform parameter a reference; valgrind flags m_tooltips as lost data, but if we try to manage it ourselves, many more leaks occur.

Global `seq64::mainwnd::on_key_press_event (GdkEventKey *a_ev)`

Test this functionality in old and new application.

Global `seq64::mainwnd::on_key_release_event (GdkEventKey *a_ev)`

Test this functionality in old and new application.

Global `seq64::mainwnd::open_performance_edit ()`

Try to find a way to set m_modified only if the song editor actually changes something, instead of just because it was opened.

Global `seq64::perfedit::perfedit (perform &p, int ppqn=SEQ64_USE_DEFAULT_PPQN, int bpm=DEFAULT_BEATS_PER_MEASURE, int bw=DEFAULT_BEAT_WIDTH)`

Offload most of the work into an initialization function like options does; make the perform parameter a reference.

Global `seq64::perform::set_bpm (int 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 `seq64::seqedit::get_measures ()`

Create a sequence::set_units() function or a sequence::get_measures() function to forward to.

Global `seq64::seqedit::seqedit (sequence &a_seq, perform &a_perf, int pos, int ppqn=SEQ64_USE_DEFAULT_PPQN)`

Offload most of the work into an initialization function like options does; make the sequence and perform parameters references.

Global [seq64::seqedit::set_background_sequence](#) (int a_seq)

Make the sequence pointer a reference.

Global [seq64::seqmenu::seq_clear_perf](#) ()

All of [seq_paste\(\)](#) can be offloaded to a (new) perform member function.

Global [seq64::seqmenu::seq_copy](#) ()

Can be offloaded to a perform member function that accepts a sequence clipboard non-const reference parameter.

Global [seq64::seqmenu::seq_cut](#) ()

A lot of [seq_cut\(\)](#) can be offloaded to a (new) perform member function that takes a sequence clipboard non-const reference parameter.

Global [seq64::seqmenu::seq_paste](#) ()

All of [seq_paste\(\)](#) can be offloaded to a (new) perform member function with a const clipboard reference parameter.

Global [seq64::sequence::remove](#) (event *e)

Use find instead in [sequence::remove\(\)](#)!

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::AbstractPerfInput	21
seq64::Seq24PerfInput	120
seq64::click	21
seq64::configfile	24
seq64::optionsfile	88
seq64::userfile	180
seq64::event	25
seq64::event_list::event_key	30
seq64::event_list	31
seq64::font	35
seq64::gui_assistant	37
seq64::gui_assistant_gtk2	39
seq64::gui_palette_gtk2	42
seq64::gui_drawingarea_gtk2	40
seq64::maintime	59
seq64::mainwid	61
seq64::perfnames	94
seq64::perfroll	111
seq64::perftime	115
seq64::seqdata	123
seq64::seqevent	133
seq64::seqkeys	138
seq64::seqroll	144
seq64::seqtime	148
seq64::gui_window_gtk2	43
seq64::mainwnd	67
seq64::perfeddit	90
seq64::seqedit	127
seq64::jack_assistant	45
seq64::jack_scratchpad	47
seq64::keybindentry	47
seq64::keys_perform	49
seq64::keys_perform_gtk2	53
seq64::keys_perform_transfer	55
seq64::keystroke	55
seq64::lash	58

seq64::midi_container	74
seq64::midi_list	76
seq64::midi_vector	78
seq64::midifile	80
seq64::options	87
seq64::perform	97
seq64::performcallback	110
seq64::mainwnd	67
rc_settings	118
seq64::gui_drawingarea_gtk2::rect	120
seq64::rect	120
seq64::Seq24SeqEventInput	122
seq64::Seq24SeqRollInput	122
seq64::seqmenu	141
seq64::mainwid	61
seq64::perfnames	94
seq64::sequence	150
seq64::trigger	167
user_instrument	168
user_instrument_t	171
user_midi_bus	171
user_midi_bus_t	173
user_settings	173

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

seq64::AbstractPerfInput	Provides an abstract base class to provide the minimal interface for the various "perf input" classes	??
seq64::click	Encapsulates any possible mouse click	??
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::event_key	Provides a key value for an event map	??
seq64::event_list	Receptable for MIDI events	??
seq64::font	This class provides a wrapper for rendering fonts that are encoded as a 16 x 16 pixmap file in XPM format	??
seq64::gui_assistant	This class provides an interface for some of the GUI support needed in Sequencer64	??
seq64::gui_assistant_gtk2	This class provides an interface for some of the Gtk/Gdk/Glib support needed in Sequencer64	??
seq64::gui_drawingarea_gtk2	Implements the basic drawing areas of the application	??
seq64::gui_palette_gtk2	Implements a stock palette of Gdk::Color elements	??
seq64::gui_window_gtk2	This class supports a basic interface for Gtk::Window-derived objects	??
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::keybindentry	Class for management of application key-bindings	??
seq64::keys_perform	This class supports the performance mode	??
seq64::keys_perform_gtk2	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::maintime	This class provides the drawing of the progress bar at the top of the main window, along with the "pills" that move in time with the measures	??
seq64::mainwid	This class implement the piano roll area of the application	??
seq64::mainwnd	This class implements the functionality of the main window of the application, except for the Patterns Panel functionality, which is implemented in the mainwid class	??
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::options	This class supports a full tabbed options dialog	??
seq64::optionsfile	Provides a file for reading and writing the application' main configuration file	??
seq64::perfedit	This class supports a Performance Editor that is used to arrange the patterns/sequences defined in the patterns panel, I think	??
seq64::perfnames	This class implements the left-side keyboard in the patterns window	??
seq64::perform	This class supports the performance mode	??
seq64::performcallback	Provides for notification of events	??
seq64::perfroll	This class implements the performance roll user interface	??
seq64::perftime	This class implements drawing the piano time at the top of the "performance window" (the "song editor")	??
rc_settings	This class contains the options formerly named "global_XXXXXX"	??
seq64::gui_drawingarea_gtk2::rect	A small helper structure representing a rectangle	??
seq64::rect	A small helper class representing a rectangle	??
seq64::Seq24PerfInput	Implements the default performance input characteristics of this application	??
seq64::Seq24SeqEventInput	This structure implement the normal interaction methods for Seq24	??
seq64::Seq24SeqRollInput	Implements the Seq24 mouse interaction paradigm for the seqroll	??
seq64::seqdata	This class supports drawing piano-roll events on a window	??
seq64::seqedit	Implements the Pattern Editor, which has references to:	??

seq64::seqevent	Implements the piano event drawing area	??
seq64::seqkeys	This class implements the left side piano of the pattern/sequence editor	??
seq64::seqmenu	This class handles the right-click menu of the sequence slots in the pattern window	??
seq64::seqroll	Implements the piano roll section of the pattern editor	??
seq64::seqtime	This class implements the piano time, whatever that is	??
seq64::sequence	Firstly a receptacle 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 se- quences and the configuration of the user-interface	??
seq64::userfile	Supports the user's ~/.seq24usr configuration file	??

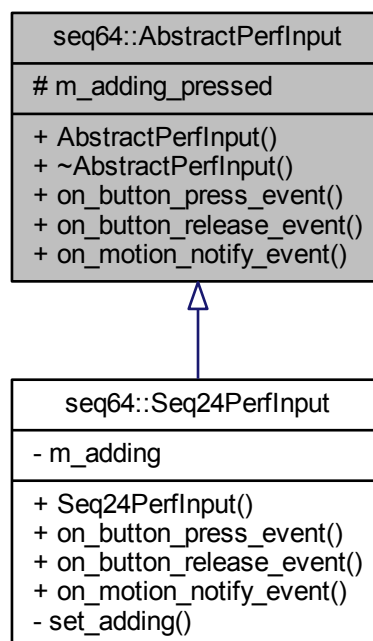
Chapter 7

Data Structure Documentation

7.1 seq64::AbstractPerfInput Class Reference

Provides an abstract base class to provide the minimal interface for the various "perf input" classes.

Inheritance diagram for seq64::AbstractPerfInput:



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](#)=SEQ64_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 the left button.
- bool [is_middle](#) () const
'Getter' function for member [m_button](#) to test for the middle button.
- bool [is_right](#) () const
'Getter' function for member [m_button](#) to test for the right button.
- 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.

Private Attributes

- bool [m_is_press](#)
Determines if the click was a press or a release event.
- int [m_x](#)
The x-coordinate of the click.
- int [m_y](#)
The y-coordinate of the click.
- int [m_button](#)
The button that was pressed or released.
- seq_modifier_t [m_modifier](#)
The optional modifier value.

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

<i>x</i>	The putative x value of the button click.
<i>y</i>	The putative y value of the button click.
<i>button</i>	The value of the button that was clicked, set to 1, 2, or 3.
<i>press</i>	Set to true if the event was a button press, false if it was a button release.
<i>modkey</i>	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

<i>rhs</i>	Provides 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

<i>rhs</i>	Provides the source object to be assigned from. The assignment is not made if "this" has the same address as this parameter.
------------	--

7.2.4 Field Documentation

7.2.4.1 int seq64::click::m_x [private]

0 is the left-most coordinate.

7.2.4.2 int seq64::click::m_y [private]

0 is the top-most coordinate.

7.2.4.3 int seq64::click::m_button [private]

Left is 1, middle is 2, and right is 3. These numbers are defined via macros, and a Linux-specific and Gtk-specific.

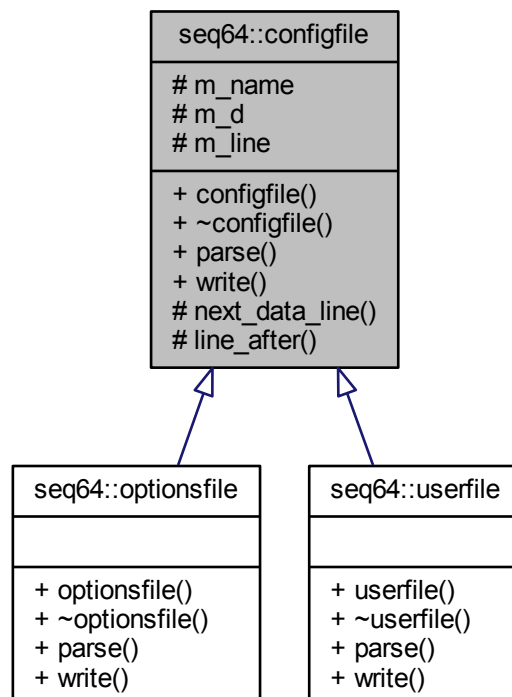
7.2.4.4 seq_modifier_t seq64::click::m_modifier [private]

Note that SEQ64_NO_MASK is our word for 0, meaning "no modifier".

7.3 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 destructor 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.

7.3.1 Constructor & Destructor Documentation

7.3.1.1 seq64::configfile::configfile (const std::string & name)

Parameters

<i>name</i>	The name of the configuration file.
-------------	-------------------------------------

7.3.2 Member Function Documentation

7.3.2.1 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

<i>a_file</i>	Points to an input stream.
---------------	----------------------------

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

Parameters

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

7.3.3 Field Documentation

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

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

7.4 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.

Private Attributes

- unsigned char `m_status`
This is status byte without the channel.
- unsigned char `m_data` [MIDI_DATA_BYTE_COUNT]
The two bytes of data for the MIDI event.
- unsigned char * `m_sysex`
Points to the data buffer for SYSEX messages.
- long `m_size`

- Gives the size of the SYSEX message.*

 - `event * m_linked`
This event is used to link Note Ons and Offs together.
- `bool m_has_link`
Indicates that a link has been made.
- `bool m_selected`
Answers the question "is this event selected in editing.".
- `bool m_marked`
Answers the question "is this event marked in processing.".
- `bool m_painted`
Answers the question "is this event being painted.".

7.4.1 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.

7.4.2 Member Function Documentation

7.4.2.1 `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

<i>rhs</i>	The object to be compared against.
------------	------------------------------------

Returns

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

7.4.2.2 `void seq64::event::mod_timestamp (unsigned long a_mod) [inline]`

Parameters

<i>a_mod</i>	The value to mod the timestamp against.
--------------	---

Returns

Returns a value ranging from 0 to *a_mod*-1.

7.4.2.3 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..

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

Parameters

<i>d1</i>	The byte value to set. We should make these all "midibytes".
-----------	--

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

Parameters

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

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

Parameters

<i>d0</i>	[out] The return reference for the first byte.
<i>d1</i>	[out] The return reference for the first byte.

7.4.2.7 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

<i>a_data</i>	Provides the additional SYSEX data.
<i>a_size</i>	Provides the size of the additional SYSEX data.

Returns

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

7.4.2.8 `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.

7.4.3 Field Documentation

7.4.3.1 `unsigned char seq64::event::m_status [private]`

The channel will be appended on the MIDI bus. The high nibble = type of event; The low nibble = channel. Bit 7 is present in all status bytes.

7.4.3.2 `unsigned char seq64::event::m_data[MIDI_DATA_BYTE_COUNT] [private]`

Remember that the most-significant bit of a data byte is always 0.

7.4.3.3 `unsigned char* seq64::event::m_sysex [private]`

This really ought to be a Boost or STD scoped pointer.

7.4.3.4 `bool seq64::event::m_has_link [private]`

This item is used [via the `get_link()` and `link()` accessors] in the sequence class.

7.5 `seq64::event_list::event_key` Class Reference

Provides a key value for an event map.

Public Member Functions

- `event_key` (unsigned long tstamp, int rank)
Principal `event_key` constructor.
- `event_key` (const `event` &e)
Event-based constructor.
- `bool operator<` (const `event_key` &rhs) const
Provides the minimal operator needed to sort events using an `event_key`.

Private Attributes

- unsigned long `m_timestamp`
The primary key-value for the key.
- int `m_rank`
The sub-key-value for the key.

7.5.1 Detailed Description

Its types match the `m_timestamp` and `get_rank()` function of this event class.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 seq64::event_list::event_key::event_key (unsigned long *tstamp*, int *rank*)

Parameters

<i>tstamp</i>	The time-stamp is the primary part of the key. It is the most important key item.
<i>rank</i>	Rank is an arbitrary number used to prioritize events that have the same time-stamp. See the event::get_rank() function for more information.

7.5.2.2 seq64::event_list::event_key::event_key (const event & *rhs*)

This constructor makes it even easier to create an [event_key](#). Note that the call to [event::get_rank\(\)](#) makes a simple calculation based on the status of the event.

Parameters

<i>rhs</i>	Provides the event key to be copied.
------------	--------------------------------------

7.5.3 Member Function Documentation

7.5.3.1 bool seq64::event_list::event_key::operator< (const event_key & *rhs*) const

Parameters

<i>e</i>	Provides the event key to be compared against.
----------	--

7.5.4 Field Documentation

7.5.4.1 unsigned long seq64::event_list::event_key::m_timestamp [private]

7.5.4.2 int seq64::event_list::event_key::m_rank [private]

7.6 seq64::event_list Class Reference

The [event_list](#) class is a receptable for MIDI events.

Data Structures

- class [event_key](#)
Provides a key value for an event map.

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 merge (event_list &el, bool presort=true)`
Provides a merge operation for the event multimap analogous to the merge operation for the event list.
- `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.

Private Types

- `typedef std::multimap< event_key, event > Events`
Types to use to swap between list and multimap implementations.

Private Member Functions

- `void link_new ()`
Links a new event.
- `void clear_links ()`
Clears all event links and unmarks them all.
- `void verify_and_link (long slength)`
This function verifies state: all note-ons have an off, and it links note-offs with their note-ons.
- `void mark_selected ()`
Marks all selected events.
- `void mark_out_of_range (long slength)`
Marks all events that have a time-stamp that is out of range.
- `void unmark_all ()`

- Unmarks all events.*
- void `unpaint_all` ()
 - Unpaints all list-events.*
- int `count_selected_notes` ()
 - Counts the selected note-on events in the event list.*
- int `count_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 `unselect_all` ()
 - Deselects all events, unconditionally.*
- void `print` ()
 - Prints a list of the currently-held events.*
- const `Events` & `events` () const
 - 'Getter' function for member m_events*

Private Attributes

- `Events m_events`
 - This list holds the current pattern/sequence events.*

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

<i>rhs</i>	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

<i>rhs</i>	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

<i>e</i>	Provides the event to be added to the list.
<i>postsort</i>	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.

7.6.3.4 void seq64::event_list::merge (event_list & el, bool *presort* = true)

We have certain constraints to preserve, as the following discussion shows.

For std::list, sequence merges list T into list A by first calling T.sort(), and then A.merge(T). The [merge\(\)](#) operation merges T into A by transferring all of its elements, at their respective ordered positions, into A. Both containers must already be ordered.

The merge effectively removes all the elements in T (which becomes empty), and inserts them into their ordered position within container (which expands in size by the number of elements transferred). The operation is performed without constructing nor destroying any element, whether T is an lvalue or an rvalue, or whether the value-type supports move-construction or not.

Each element of T is inserted at the position that corresponds to its value according to the strict weak ordering defined by operator <. The resulting order of equivalent elements is stable (i.e. equivalent elements preserve the relative order they had before the call, and existing elements precede those equivalent inserted from x). The function does nothing if (&x == this).

For std::multimap, sorting is automatic. However, unless move-construction is supported, merging will be less efficient than for the list version. Also, we need a way to include duplicates of each event, so we need to use a multimap. Once all this setup, merging is really just insertion. And, since sorting isn't needed, the multimap actually turns out to be faster.

Parameters

<i>el</i>	Provides the event list to be merged into the current event list.
<i>presort</i>	If true, the events are presorted. This is a requirement for merging an std::list, but is a no-op for the std::multimap implementation.

7.6.3.5 void seq64::event_list::link_new () [private]

This function checks for a note on, then look for its note off. This function is provided in the [event_list](#) because it does not depend on any external data. Also note that any desired thread-safety must be provided by the caller.

7.6.3.6 void seq64::event_list::verify_and_link (long *length*) [private]

Threadsafe

Parameters

<i>slength</i>	Provides the length beyond which events will be pruned.
----------------	---

7.6.3.7 void seq64::event_list::mark_out_of_range (long *slength*) [private]

Used for killing (pruning) those events not in range. If the current time-stamp is greater than the length, then the event is marked for pruning.

Parameters

<i>slength</i>	Provides the length beyond which events will be pruned.
----------------	---

7.6.3.8 int seq64::event_list::count_selected_events (unsigned char *status*, unsigned char *cc*) [private]

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

7.7 seq64::font Class Reference

This class provides a wrapper for rendering fonts that are encoded as a 16 x 16 pixmap file in XPM format.

Public Types

- enum [Color](#) {
[BLACK](#),
[WHITE](#),
[BLACK_ON_YELLOW](#),
[YELLOW_ON_BLACK](#) }

Public Member Functions

- [font](#) ()
Rotate default constructor.
- void [init](#) (Glib::RefPtr< Gdk::Window > a_window)
Initialization function for a window on which fonts will be drawn.
- void [render_string_on_drawable](#) (Glib::RefPtr< Gdk::GC > m_gc, int x, int y, Glib::RefPtr< Gdk::Drawable > a_draw, const char *str, [font::Color](#) col)
Draws a text string.

Private Attributes

- Glib::RefPtr< Gdk::Pixmap > * [m_pixmap](#)
Points to the current pixmap (m_black_pixmap or m_white_pixmap) to use to render a string.
- Glib::RefPtr< Gdk::Pixmap > [m_black_pixmap](#)
The pixmap in the file src/pixmaps/font_b.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_white_pixmap](#)
The pixmap in the file src/pixmaps/font_b.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_b_on_y_pixmap](#)
The pixmap in the file src/pixmaps/font_y.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_y_on_b_pixmap](#)

The pixmap in the file `src/pixmaps/font_yb.xpm` is loaded into this object.

- `Glib::RefPtr<Gdk::Bitmap>` `m_clip_mask`

This object is instantiated as a default object.

7.7.1 Member Enumeration Documentation

7.7.1.1 enum `seq64::font::Color`

Enumerator

BLACK A simple enumeration to describe the basic colors used in writing text. Basically, these two values cause the selection of one or another pixmap (`font_b_xpm` and `font_w_xpm`). We've added two more pixmaps to draw black text on a yellow background (`font_y.xpm`) and yellow text on a black background (`font_yb.xpm`).

The first supported color. A black font on a white background.

WHITE The second supported color. A white font on a black background.

BLACK_ON_YELLOW A new color, for drawing black text on a yellow background.

YELLOW_ON_BLACK A new color, for drawing yellow text on a black background.

7.7.2 Member Function Documentation

7.7.2.1 void `seq64::font::init` (`Glib::RefPtr<Gdk::Window>` `a_window`)

This function loads two pixmaps that contain the characters to be used to draw text strings. Both pixmaps provide a 16 x 16 grid of boxes, and each box contains one of the 256 characters in this font set.

One pixmap has white characters on a black background, and other other has black characters on a white background. See the descriptions of the `c_text_x` and `c_text_y` variables in the `globals` module.

7.7.2.2 void `seq64::font::render_string_on_drawable` (`Glib::RefPtr<Gdk::GC>` `a_gc`, `int x`, `int y`, `Glib::RefPtr<Gdk::Drawable>` `a_draw`, `const char *` `str`, `font::Color col`)

This function grabs the proper font bitmap, extracts the current character pixmap from it, and slaps it down where it needs to be to render the character in the string.

Parameters

<code>a_gc</code>	Provides the graphics context for drawing the text using GTK+.
<code>x</code>	The horizontal location of the text.
<code>y</code>	The vertical location of the text.
<code>a_draw</code>	The drawable object on which to draw the text.
<code>str</code>	The string to draw. Should use a constant string reference instead.
<code>col</code>	The font color to use to draw the string. The only support values are <code>font::BLACK</code> and <code>font::WHITE</code> , and the correct colors are provided by selecting one of two font pixmaps, as described in the <code>init()</code> function.

7.7.3 Field Documentation

7.7.3.1 `Glib::RefPtr<Gdk::Pixmap> * seq64::font::m_pixmap` `[private]`

This member used to be an object, but it's probably a bit faster to just use a pointer (or a reference).

7.7.3.2 `Glib::RefPtr<Gdk::Pixmap> seq64::font::m_black_pixmap` `[private]`

It contains a black font on a white background.

7.7.3.3 `Glib::RefPtr<Gdk::Pixmap> seq64::font::m_white_pixmap` `[private]`

It contains a black font on a white background.

7.7.3.4 `Glib::RefPtr<Gdk::Pixmap> seq64::font::m_b_on_y_pixmap` `[private]`

It contains a black font on a yellow background.

7.7.3.5 `Glib::RefPtr<Gdk::Pixmap> seq64::font::m_y_on_b_pixmap` `[private]`

It contains a yellow font on a black background.

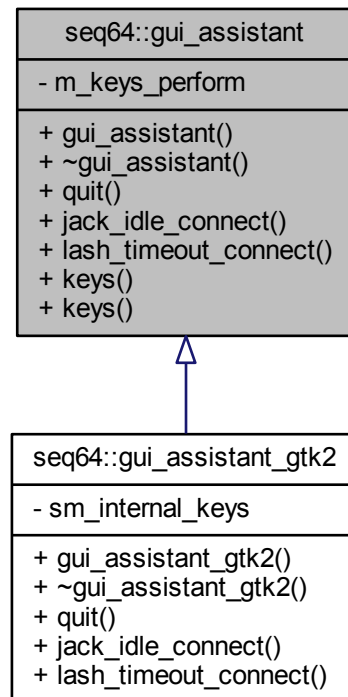
7.7.3.6 `Glib::RefPtr<Gdk::Bitmap> seq64::font::m_clip_mask` `[private]`

All we know is it seems to be a requirement for creating a pixmap object from an XMP file.

7.8 seq64::gui_assistant Class Reference

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

Inheritance diagram for seq64::gui_assistant:



Public Member Functions

- [gui_assistant](#) ([keys_perform](#) &kp)
This constructor wires in some externally (for now) created objects.
- virtual [~gui_assistant](#) ()
Stock base-class implementation of a virtual destructor.
- 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.

Private Attributes

- [keys_perform](#) & [m_keys_perform](#)
Provides a reference to the app-specific GUI-specific keys_perform-derived object that an application is going to use for handling sequence-control keys.

7.8.1 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).

7.8.2 Constructor & Destructor Documentation

7.8.2.1 seq64::gui_assistant::gui_assistant (keys_perform & kp)

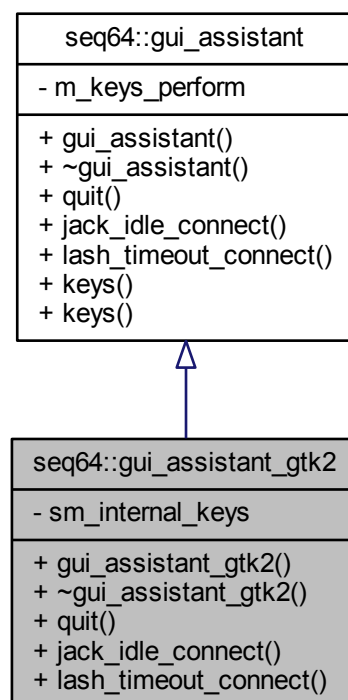
Parameters

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

7.9 seq64::gui_assistant_gtk2 Class Reference

This class provides an interface for some of the Gtk/Gdk/Glib support needed in Sequencer64.

Inheritance diagram for seq64::gui_assistant_gtk2:



Public Member Functions

- [gui_assistant_gtk2 \(\)](#)
This class provides an interface for some of the Gtk/Gdk/Glib support needed in Sequencer64.
- virtual void [quit \(\)](#)
Calls the Glib Main object's [quit\(\)](#) function.
- virtual void [jack_idle_connect](#) ([jack_assistant](#) &jack)
Connects the JACK session-event callback to the Glib idle object.
- virtual void [lash_timeout_connect](#) ([lash](#) &lashobject)
Connects the LASH timeout-event callback to the Glib timeout object.

Static Private Attributes

- static keys_perform_gtk2 sm_internal_keys

Provides a pre-made `keys_perform` object.

7.9.1 Field Documentation

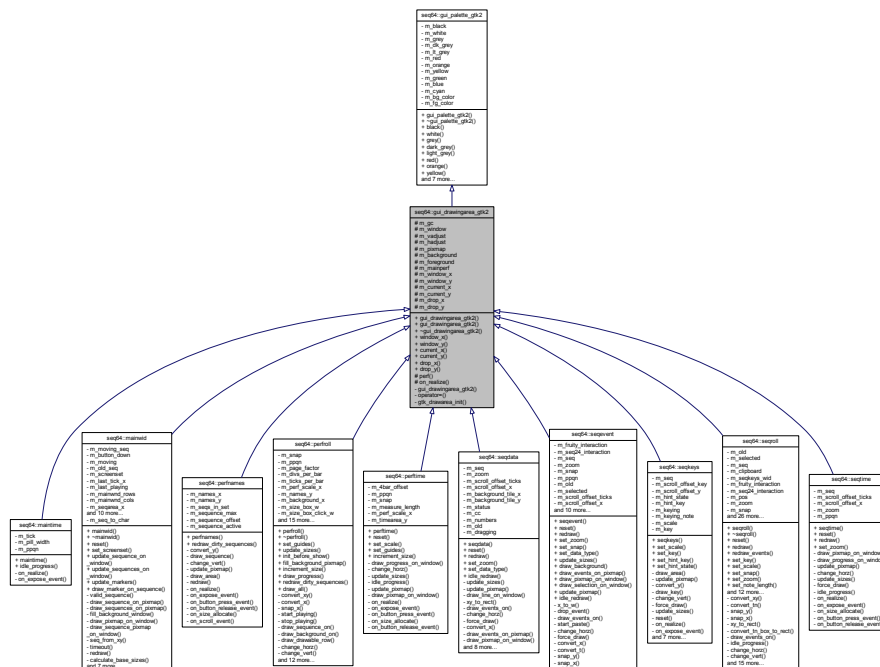
7.9.1.1 keys_perform_gtk2 seq64::gui_assistant_gtk2::sm_internal_keys [static],[private]

This object is set into the reference provided in the [gui_assistant](#) base class.

7.10 seq64::gui_drawingarea_gtk2 Class Reference

Implements the basic drawing areas of the application.

Inheritance diagram for seq64::gui_drawingarea_gtk2:



Data Structures

- struct rect

A small helper structure representing a rectangle.

Public Member Functions

- `gui_drawingarea_gtk2` (perform &p, int window_x=0, int window_y=0)

Perform-only constructor.

- `gui_drawingarea_gtk2` (perform &a_perf, Gtk::Adjustment &a_hadjust, Gtk::Adjustment &a_vadjust, int window x=0, int window y=0)

Principal constructor.

- [~gui_drawingarea_gtk2](#) ()
Provides a destructor to delete allocated objects.
- int [window_x](#) () const
'Getter' function for member m_window_x
- int [window_y](#) () const
'Getter' function for member m_window_y
- int [current_x](#) () const
'Getter' function for member m_current_x
- int [current_y](#) () const
'Getter' function for member m_current_y
- int [drop_x](#) () const
'Getter' function for member m_drop_x
- int [drop_y](#) () const
'Getter' function for member m_drop_y

Protected Member Functions

- [perform](#) & [perf](#) ()
'Getter' function for member m_mainperf
- void [on_realize](#) ()
For this GTK callback, on realization of window, initialize the shiz.

Protected Attributes

- [perform](#) & [m_mainperf](#)
A frequent hook into the main perform object.
- int [m_window_x](#)
Window sizes.
- int [m_current_x](#)
The x and y value of the current location of the mouse (during dragging?)
- int [m_drop_x](#)
These values are used when roping and highlighting a bunch of events.

Private Member Functions

- void [gtk_drawarea_init](#) ()
Does basic initialization for each of the constructors.

Additional Inherited Members

7.10.1 Detailed Description

Note that this class really "isn't a" `gui_palette_gtk2`; it should simply have one. But that base class must be derived from `Gtk::DrawingArea`. We don't want to waste some space by using a "has-a" relationship, and also put up with having to access the palette indirectly. So, in this case, we tolerate the less strict implementation.

7.10.2 Member Function Documentation

7.10.2.1 void seq64::gui_drawingarea_gtk2::on_realize () [protected]

It allocates any additional resources that weren't initialized in the constructor.

Protected Types

- typedef Gdk::Color [Color](#)

Provides a type for the color object.

7.11.1 Detailed Description

Note that this class must be derived from `Gtk::DrawingArea` (or `Gtk::Widget`) in order to get access to the `get_↔default_colormap()` function used in the constructor.

7.11.2 Constructor & Destructor Documentation

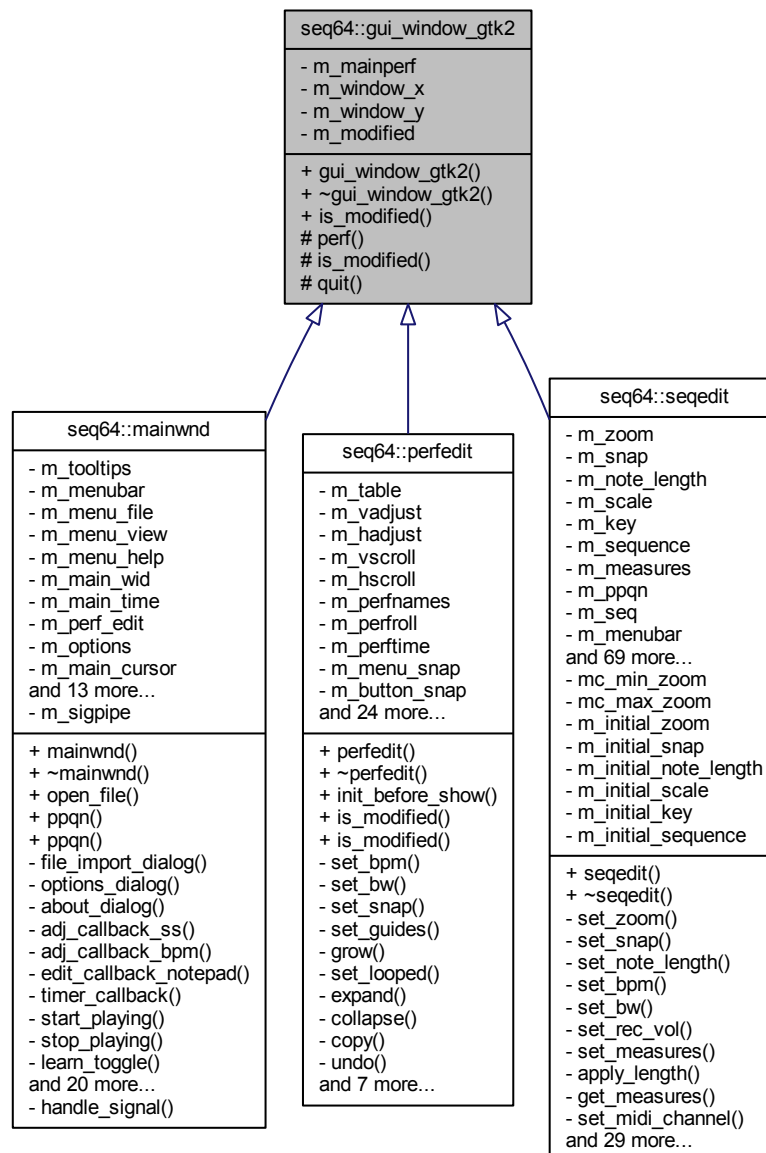
7.11.2.1 seq64::gui_palette_gtk2::gui_palette_gtk2 ()

In the constructor you can only allocate colors; `get_window()` returns 0 because this window has not be realized.

7.12 seq64::gui_window_gtk2 Class Reference

This class supports a basic interface for `Gtk::Window`-derived objects.

Inheritance diagram for seq64::gui_window_gtk2:



Public Member Functions

- [gui_window_gtk2](#) ([perform](#) &p, int window_x=0, int window_y=0)

Principal constructor, has a reference to the all-important perform object.

- [~gui_window_gtk2](#) ()

This rote constructor does nothing.

- bool [is_modified](#) () const

'Getter' function for member m_modified

Protected Member Functions

- `perform` & `perf` ()
'Getter' function for member m_mainperf
- `void is_modified` (bool flag)
'Setter' function for member m_modified

Private Attributes

- `int m_window_x`
Window sizes.

7.12.1 Constructor & Destructor Documentation

7.12.1.1 `seq64::gui_window_gtk2::gui_window_gtk2 (perform & p, int window_x = 0, int window_y = 0)`

Parameters

<code>a_perf</code>	Refers to the main performance object.
---------------------	--

7.12.2 Field Documentation

7.12.2.1 `int seq64::gui_window_gtk2::m_window_x` [`private`]

Could make this constant, but some windows are resizable.

7.13 seq64::jack_assistant Class Reference

This class provides the performance mode JACK support.

Public Member Functions

- `jack_assistant` (`perform` & `parent`, `int ppqn`=SEQ64_USE_DEFAULT_PPQN)
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.

7.13.1 Constructor & Destructor Documentation

7.13.1.1 `seq64::jack_assistant::jack_assistant (perform &parent, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

Parameters

<code>parent</code>	Provides a reference to the main perform object that needs to control JACK event.
---------------------	---

7.13.2 Member Function Documentation

7.13.2.1 `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.)

7.13.2.2 `void seq64::jack_assistant::position (bool a_state)`

Warning

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

Parameters

<code>state</code>	If true, the current tick is set to the leftmost tick.
--------------------	--

7.13.2.3 `bool seq64::jack_assistant::output (jack_scratchpad &pad)`

Parameters

<i>pad</i>	Provide a JACK scratchpad, whatever that is.
------------	--

Returns

Returns true if JACK is running.

7.13.3 Friends And Related Function Documentation

7.13.3.1 `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

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

7.13.3.2 `void jack_shutdown (void * arg)` [*friend*]

Parameters

<i>arg</i>	Points to the jack_assistant in charge of JACK support for the perform object.
------------	--

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

Parameters

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

7.14 seq64::jack_scratchpad Struct Reference

Provide a temporary structure for passing data and results between a perform and [jack_assistant](#) object.

7.14.1 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\(\)](#).

7.15 seq64::keybindentry Class Reference

Class for management of application key-bindings.

Inherits Entry.

Public Member Functions

- `keybindentry` (`type` `t`, unsigned int `*location_to_write=NULLPTR`, `perform` `*p=NULLPTR`, long `s=0`)
This constructor initializes the member with values dependent on the value type provided in the first parameter.
- void `set` (unsigned int `val`)
Gets the key name from the integer value; if there is one, then it is printed into a temporary buffer, otherwise the value is printed into that buffer as is.
- virtual bool `on_key_press_event` (GdkEventKey `*event`)
Handles a key press by calling `set()` with the event's key value.

Private Types

- enum `type` {
`location`,
`events`,
`groups` }

Private Attributes

- unsigned int `* m_key`
Points to the value of the key that is part of this key-binding.
- `type m_type`
Stores the type of key-binding.
- `perform * m_perf`
Stores an optional pointer to a perform object.
- long `m_slot`
Provides???

7.15.1 Member Enumeration Documentation

7.15.1.1 `enum seq64::keybindentry::type` [private]

Enumerator

location Provides the type of keybindings that can be made. Used for handling a keystroke made while a keyboard-options field is active, for selecting a key via the keyboard, and binding to pattern/sequence boxes, we think. It is used in the options class to associate a key with the binding.

events Used for binding to events.

groups Used for binding to groups.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `seq64::keybindentry::keybindentry (type t, unsigned int * location_to_write = NULLPTR, perform * p = NULLPTR, long s = 0)`

Usage In options, a pointer to a new key-binding entry is managed by calling `keybindentry (keybindentry←::location, &perf->keyname)`.

Parameters

<i>t</i>	Provides the type of key-binding: location, events, or groups.
<i>location_to_write</i>	The location that holds the value of the key associated with the key-binding. The default value of this parameter is the null pointer.
<i>p</i>	Points to the performance object used with this key-binding. The default value of this parameter is the null pointer.
<i>s</i>	Provides the slot value for this key-binding. The default value of this parameter is zero.

7.15.3 Member Function Documentation

7.15.3.1 void seq64::keybindentry::set (unsigned int *val*)

Then we call set_text(buf). The set_width_char() function is then called.

7.15.3.2 bool seq64::keybindentry::on_key_press_event (GdkEventKey * *event*) [virtual]

This value is used to set the event or key depending on the value of m_type.

7.15.4 Field Documentation

7.15.4.1 unsigned int* seq64::keybindentry::m_key [private]

Not yet sure by the address of this key value is needed. It can be a null pointer, as well.

7.16 seq64::keys_perform Class Reference

This class supports the performance mode.

Inheritance diagram for seq64::keys_perform:



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.

Private Attributes

- unsigned int [m_key_bpm_up](#)
Provides key assignments for some key sequencer features.

7.16.1 Detailed Description

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

7.16.2 Constructor & Destructor Documentation

7.16.2.1 seq64::keys_perform::~~keys_perform ()

Finally, any active patterns/sequences are deleted.

7.16.3 Member Function Documentation

7.16.3.1 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

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

7.16.3.2 void seq64::keys_perform::get_keys (keys_perform_transfer & kpt)

Parameters

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

7.16.3.3 bool seq64::keys_perform::show_ui_sequence_key () const [inline]

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

7.16.3.4 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

<i>key</i>	Provides the numeric value of the keystroke.
------------	--

Returns

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

Reimplemented in [seq64::keys_perform_gtk2](#).

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

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

Reimplemented in [seq64::keys_perform_gtk2](#).

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

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

Reimplemented in [seq64::keys_perform_gtk2](#).

7.16.3.7 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

<i>keycode</i>	The key to be assigned.
<i>sequence_slot</i>	The perform event slot into which the keycode will be assigned.

7.16.3.8 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

<i>keycode</i>	The key to be assigned.
<i>group_slot</i>	The perform group slot into which the keycode will be assigned.

7.16.4 Field Documentation

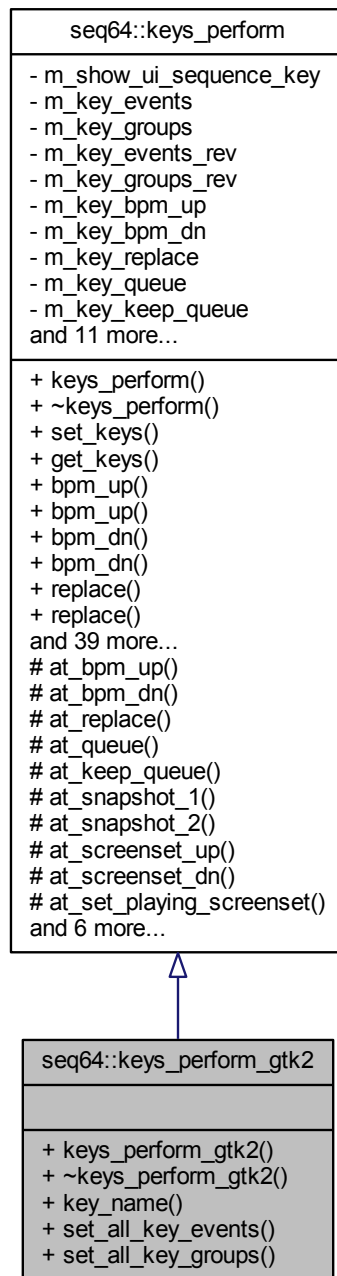
7.16.4.1 unsigned int seq64::keys_perform::m_key_bpm_up [private]

Used in mainwnd, options, optionsfile, perfedit, seqroll, userfile, and perform.

7.17 seq64::keys_perform_gtk2 Class Reference

This class supports the performance mode.

Inheritance diagram for seq64::keys_perform_gtk2:



Public Member Functions

- [keys_perform_gtk2 \(\)](#)

This construction initializes a vast number of member variables, some of them public!

- virtual [~keys_perform_gtk2 \(\)](#)

The destructor sets some running flags to false, signals this condition, then joins the input and output threads if the were launched.

- virtual std::string [key_name](#) (unsigned int key) const
Obtains the name of the key.
- virtual void [set_all_key_events](#) ()
Sets up the keys for arming/unmuting events in the Gtk-2 environment.
- virtual void [set_all_key_groups](#) ()
Sets up the keys for group events in the Gtk-2 environment.

Additional Inherited Members

7.17.1 Detailed Description

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

7.17.2 Constructor & Destructor Documentation

7.17.2.1 seq64::keys_perform_gtk2::~~keys_perform_gtk2 () [virtual]

Finally, any active patterns/sequences are deleted.

7.17.3 Member Function Documentation

7.17.3.1 std::string seq64::keys_perform_gtk2::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.

Reimplemented from [seq64::keys_perform](#).

7.17.3.2 void seq64::keys_perform_gtk2::set_all_key_events () [virtual]

The base-class function call makes sure the the related lists are cleared before rebuilding them here.

Reimplemented from [seq64::keys_perform](#).

7.17.3.3 void seq64::keys_perform_gtk2::set_all_key_groups () [virtual]

The base-class function call makes sure the the related lists are cleared before rebuilding them here.

Reimplemented from [seq64::keys_perform](#).

7.18 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().

7.19 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_MASK))
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.

Private Attributes

- bool [m_is_press](#)
Determines if the key was a press or a release.
- unsigned int [m_key](#)
The key that was pressed or released.
- seq_modifier_t [m_modifier](#)
The optional modifier value.

7.19.1 Detailed Description

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

7.19.2 Constructor & Destructor Documentation

- 7.19.2.1 [seq64::keystroke::keystroke](#) (unsigned int *key*, bool *press* = SEQ64_KEYSTROKE_PRESS, int *modkey* = int (SEQ64_NO_MASK))

Parameters

<i>key</i>	The keystroke number of the key that was pressed or released.
<i>press</i>	If true, the keystroke action was a press, otherwise it was a release.
<i>modkey</i>	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.

7.19.2.2 seq64::keystroke::keystroke (const keystroke & rhs)

Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

7.19.3 Member Function Documentation

7.19.3.1 keystroke & seq64::keystroke::operator= (const keystroke & rhs)

Parameters

<i>rhs</i>	The object to be assigned.
------------	----------------------------

Returns

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

7.19.3.2 bool seq64::keystroke::is_letter (int ch = SEQ64_KEYSTROKE_BAD_VALUE) const

Parameters

<i>ch</i>	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

7.19.4 Field Documentation

7.19.4.1 bool seq64::keystroke::m_is_press [private]

See the SEQ64_KEYSTROKE_PRESS and SEQ64_KEYSTROKE_RELEASE readability macros.

7.19.4.2 unsigned int seq64::keystroke::m_key [private]

Generally, the extended ASCII range (0 to 255) is supported. However, Gtk-2.x/3.x will generally support the full gamut of characters defined in the gdk_basic_keys.h module. We define minimum and maximum range macros for keystrokes that are a bit generous.

7.19.4.3 seq_modifier_t seq64::keystroke::m_modifier [private]

Note that SEQ64_NO_MASK is our word for 0, meaning "no modifier".

7.20 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.
- bool [process_events](#) ()
Process LASH events.

Private Member Functions

- bool [init](#) ()
Initializes LASH support, if enabled.
- void [handle_event](#) (lash_event_t *conf)
Handle a LASH event.
- void [handle_config](#) (lash_config_t *conf)
Handle a LASH configuration item.

Private Attributes

- [perform](#) & [m_perform](#)
A hook into the single perform object in the application.

7.20.1 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.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 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

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

7.20.3 Member Function Documentation

7.20.3.1 `void seq64::lash::set_alsa_client_id (int id)`

/param *id* The ALSA client ID to be set.

7.20.3.2 `bool seq64::lash::process_events ()`

Returns

Always returns true.

7.20.3.3 `bool seq64::lash::init ()` [private]

Returns

Returns true if the LASH subsystem was able to be initialized, and a LASH client representative (*m_client*) was allocated.

7.20.3.4 `void seq64::lash::handle_event (lash_event_t* ev)` [private]

Parameters

<i>ev</i>	Provides the event to be handled.
-----------	-----------------------------------

7.20.3.5 `void seq64::lash::handle_config (lash_config_t* conf)` [private]

Currently incomplete.

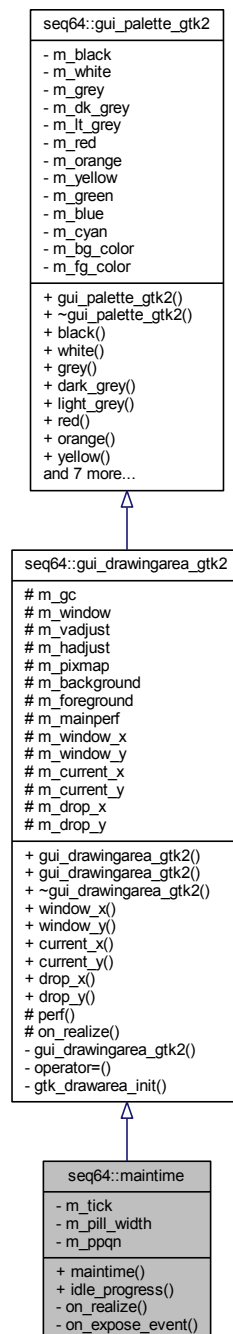
Parameters

<i>conf</i>	Provides the configuration item to handle.
-------------	--

7.21 seq64::maintime Class Reference

This class provides the drawing of the progress bar at the top of the main window, along with the "pills" that move in time with the measures.

Inheritance diagram for seq64::maintime:



Public Member Functions

- `maintime` (`perform` &p, int ppqn=SEQ64_USE_DEFAULT_PPQN, int pillwidth=c_pill_width, int x=c_maintime_x, int y=c_maintime_y)

This constructor sets up the colors black, white, and grey, and then allocates them.

- int `idle_progress` (long ticks)

This function clears the window, sets the foreground to black, draws the "time" window's rectangle, and then draws a rectangle for noting the progress of the beat, and the progress for a bar.

Private Member Functions

- void [on_realize](#) ()

Handles realization of the window.

- bool [on_expose_event](#) (GdkEventExpose *ev)

This function merely idles.

Additional Inherited Members

7.21.1 Constructor & Destructor Documentation

7.21.1.1 `seq64::maintime::maintime (perform & p, int ppqn = SEQ64_USE_DEFAULT_PPQN, int pillwidth = c_pill_width, int x = c_maintime_x, int y = c_maintime_y)`

In the constructor you can only allocate colors; `get_window()` would return 0 because the windows has not yet been realized.

7.21.2 Member Function Documentation

7.21.2.1 `int seq64::maintime::idle_progress (long ticks)`

Idle hands do the devil's work. We should eventually support some generic coloring for "dark themes". The default coloring is better for "light themes".

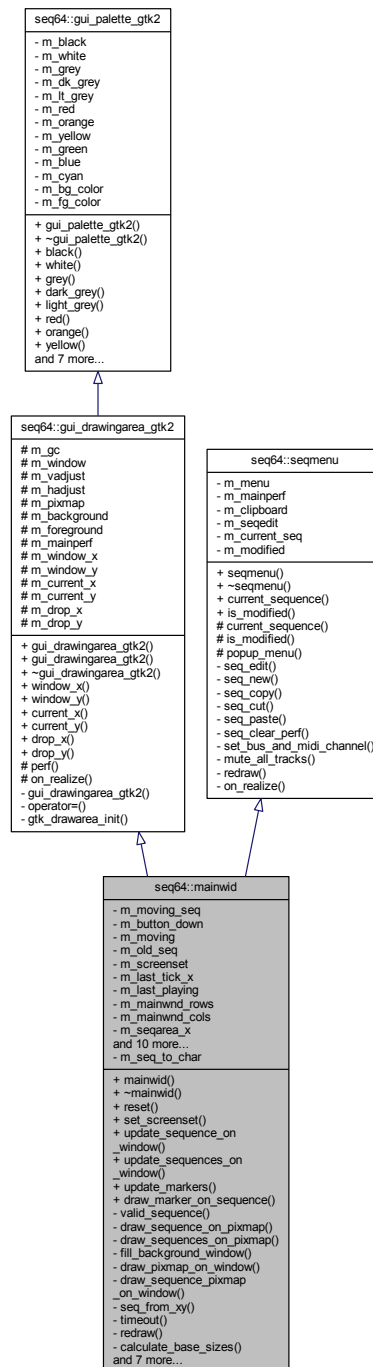
7.21.2.2 `void seq64::maintime::on_realize () [private]`

It performs the base class's [on_realize\(\)](#) function. It then allocates some additional resources: a window, a GC (?), and it clears the window. Then it sets the default size of the window, specified by GUI constructor parameters.

7.22 seq64::mainwid Class Reference

This class implement the piano roll area of the application.

Inheritance diagram for seq64::mainwid:



Public Member Functions

- [mainwid](#) (perform &p)

This constructor sets a lot of the members, but not all.

- [~mainwid](#) ()

A rote destructor.

- void [reset](#) ()

- This function redraws everything and queues up a redraw operation.*
- void [set_screenset](#) (int ss)
Set the current screen-set.
- void [update_sequence_on_window](#) (int seq)
Updates the image of one sequencer.
- void [update_sequences_on_window](#) ()
Updates the image of multiple sequencers.
- void [update_markers](#) (int ticks)
Draw the cursors (long vertical bars) on each sequence, so that they follow the playing progress of each sequence in the mainwid (Patterns Panel.)
- void [draw_marker_on_sequence](#) (int seq, int tick)
Does the actual drawing of one pattern/sequence position marker, a vertical progress bar.

Private Member Functions

- bool [valid_sequence](#) (int seq)
Common-code helper function.
- void [draw_sequence_on_pixmap](#) (int seq)
This function draws a specific pattern/sequence on the pixmap located in the main window of the application, the Patterns Panel.
- void [draw_sequences_on_pixmap](#) ()
This function fills the pixmap with sequences.
- void [fill_background_window](#) ()
This function updates the background window, clearing it.
- void [draw_pixmap_on_window](#) ()
This function queues the blit of pixmap to window.
- void [draw_sequence_pixmap_on_window](#) (int seq)
This function draws something in the Patterns Panel.
- int [seq_from_xy](#) (int x, int y)
Translates XY coordinates in the Patterns Panel to a sequence number.
- int [timeout](#) ()
Provides a stock callback, because some kind of callback is need.
- void [redraw](#) (int seq)
Draw the the given pattern/sequence again.
- void [calculate_base_sizes](#) (int seq, int &basex, int &basey)
Provides a way to calculate the base x and y size values for the pattern map.
- void [on_realize](#) ()
For this GTK callback, on realization of window, initialize the shiz.
- bool [on_expose_event](#) (GdkEventExpose *ev)
Implements the GTK expose event callback.
- bool [on_button_press_event](#) (GdkEventButton *ev)
Handles a press of a mouse button.
- bool [on_button_release_event](#) (GdkEventButton *ev)
Handles a release of a mouse button.
- bool [on_motion_notify_event](#) (GdkEventMotion *p0)
Handle the motion of the mouse if a mouse button is down and in another sequence and if the current sequence is not in edit mode.
- bool [on_focus_in_event](#) (GdkEventFocus *)
Handles an on-focus event.
- bool [on_focus_out_event](#) (GdkEventFocus *)
Handles an out-of-focus event.

Private Attributes

- int [m_mainwnd_rows](#)

These values are assigned to the values given by the constants of similar names in `globals.h`, and we will make them parameters later.

Additional Inherited Members

7.22.1 Constructor & Destructor Documentation

7.22.1.1 `seq64::mainwid::mainwid (perform & p)`

And it asks for a size of `c_mainwid_x` by `c_mainwid_y`. It adds GDK masks for button presses, releases, and motion, and key presses and focus changes.

Parameters

<code>p</code>	Provides the reference to the all-important <code>perform</code> object.
----------------	--

7.22.2 Member Function Documentation

7.22.2.1 `void seq64::mainwid::set_screenset (int a_ss)`

Parameters

<code>a_ss</code>	Provides the screen-set number to set.
-------------------	--

7.22.2.2 `void seq64::mainwid::update_sequence_on_window (int seqnum)`

Parameters

<code>seqnum</code>	Provides the number of the sequence to update.
---------------------	--

7.22.2.3 `void seq64::mainwid::update_markers (int ticks)`

Parameters

<code>ticks</code>	Starting point for drawing the markers.
--------------------	---

7.22.2.4 `void seq64::mainwid::draw_marker_on_sequence (int seqnum, int tick)`

If the sequence has no events, this function doesn't bother even drawing a position marker.

Note that, when `Sequencer64` first comes up, and [perform::is_dirty_main\(\)](#) is called, no sequences exist yet.

Parameters

<code>seqnum</code>	Provides the number of the sequence to draw.
<code>tick</code>	Provides the location to draw the marker.

7.22.2.5 `bool seq64::mainwid::valid_sequence (int seqnum) [private]`

Parameters

<i>seqnum</i>	Provides the number of the sequence to validate.
---------------	--

Returns

Returns true if the sequence number is valid for the current m_screenset value.

7.22.2.6 void seq64::mainwid::draw_sequence_on_pixmap (int *seqnum*) [private]

The sequence is drawn only if it is in the current screen set (indicated by m_screenset).

Also, we now ignore the sequence if it does not exist. :-D

Note

If only the main window is up, then the sequences just appear to play – the progress bars move in each pattern. Gaps in the song don't change the appearance of the patterns. But, if the Song (performance) Editor window is up, and the song is started using the controls in the Song (performance) Editor windows, then the active patterns are black (!) while playing, and white when gaps in the song are encountered. Also, the muting status in the main window seems to be ignored (based on coloring, anyway). However, the muting in the Song (performance) windows does seem to be in force.

Parameters

<i>seqnum</i>	Provides the number of the sequence slot that needs to be drawn.
---------------	--

7.22.2.7 void seq64::mainwid::draw_sequences_on_pixmap () [private]

Please note that [draw_sequence_on_pixmap\(\)](#) also draws the empty boxes of inactive sequences, so we cannot take shortcuts here.

7.22.2.8 void seq64::mainwid::draw_sequence_pixmap_on_window (int *seqnum*) [private]

The sequence is drawn only if it is in the current screen set (indicated by m_screenset). However, if we comment out this code, we can't see any difference in the Patterns Panel, even when playback is ongoing!

Parameters

<i>seqnum</i>	Provides the number of the sequence to draw.
---------------	--

7.22.2.9 int seq64::mainwid::seq_from_xy (int *a_x*, int *a_y*) [private]

Parameters

<i>a_x</i>	Provides the x coordinate.
<i>a_y</i>	Provides the y coordinate.

Returns

Returns -1 if the sequence number cannot be calculated.

7.22.2.10 `int seq64::mainwid::timeout () [private]`

Todo We should use this callback to display the current time in the playback.

Returns

Always returns true.

7.22.2.11 `void seq64::mainwid::redraw (int seqnum) [private], [virtual]`

Parameters

<i>seqnum</i>	Provides the number of the sequence to draw.
---------------	--

Implements [seq64::seqmenu](#).

7.22.2.12 `void seq64::mainwid::calculate_base_sizes (int seqnum, int & basex, int & basey) [private]`

The values are returned as side-effects.

Parameters

<i>seqnum</i>	Provides the number of the sequence to calculate.
<i>basex</i>	A return parameter for the x coordinate of the base size.
<i>basey</i>	A return parameter for the y coordinate of the base size.

7.22.2.13 `void seq64::mainwid::on_realize () [private]`

It allocates any additional resources that weren't initialized in the constructor.

7.22.2.14 `bool seq64::mainwid::on_expose_event (GdkEventExpose * a_e) [private]`

Parameters

<i>a_e</i>	The expose event.
------------	-------------------

Returns

Always returns true.

7.22.2.15 `bool seq64::mainwid::on_button_press_event (GdkEventButton * p0) [private]`

It grabs the focus, calculates the pattern/sequence over which the button press occurred, and sets the `m_button_↔_down` flag if it is over a pattern.

Parameters

<i>p0</i>	Provides the parameters of the button event.
-----------	--

Returns

Always returns true.

7.22.2.16 `bool seq64::mainwid::on_button_release_event (GdkEventButton * p0)` [private]

This event is a lot more complex than a press. The left button toggles playback status. The right button brings up a popup menu. If the slot is empty, then a "New" popup is presented, otherwise an "Edit" and selection popup is presented.

Parameters

<i>p0</i>	Provides the parameters of the button event.
-----------	--

Returns

Always returns true.

7.22.2.17 `bool seq64::mainwid::on_motion_notify_event (GdkEventMotion * p0)` [private]

This function moves the selected pattern to another pattern slot.

Parameters

<i>p0</i>	Provides the parameters of the button event.
-----------	--

Returns

Always returns true.

7.22.2.18 `bool seq64::mainwid::on_focus_in_event (GdkEventFocus *)` [private]

Just sets the Gtk::HAS_FOCUS flag.

Returns

Always returns false.

7.22.2.19 `bool seq64::mainwid::on_focus_out_event (GdkEventFocus *)` [private]

Just unsets the Gtk::HAS_FOCUS flag.

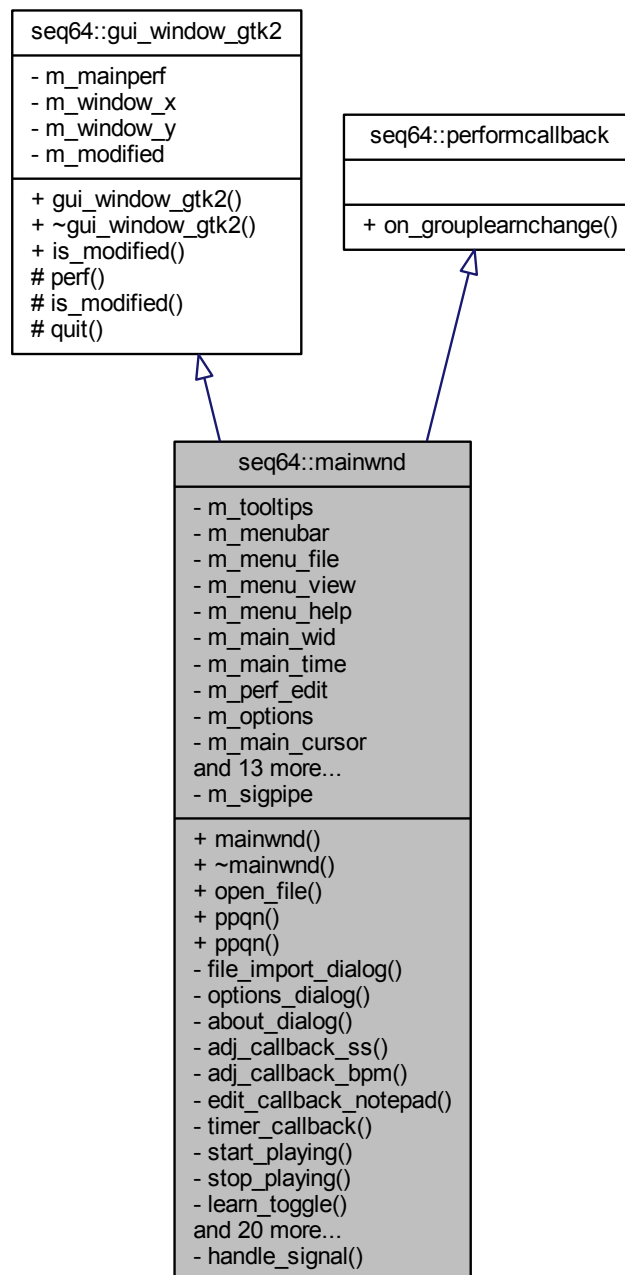
Returns

Always returns false.

7.23 seq64::mainwnd Class Reference

This class implements the functionality of the main window of the application, except for the Patterns Panel functionality, which is implemented in the mainwid class.

Inheritance diagram for seq64::mainwnd:



Public Member Functions

- `mainwnd` (`perform` &`a_p`)

The constructor the main window of the application.

- `~mainwnd` ()

This destructor must explicitly delete some allocated resources.

- void `open_file` (const std::string &)

- *Opens and parses (reads) a MIDI file.*
- int `ppqn` () const
'Getter' function for member m_ppqn
- void `ppqn` (int ppqn)
'Setter' function for member m_ppqn We can't set the PPQN value when the mainwnd is created, we have to do it later, using this function.

Private Member Functions

- void `file_import_dialog` ()
Presents a file dialog to import a MIDI file.
- void `options_dialog` ()
Opens the File / Options dialog.
- void `about_dialog` ()
Presents a Help / About dialog.
- void `adj_callback_ss` ()
This function is the callback for adjusting the screen-set value.
- void `adj_callback_bpm` ()
This function is the callback for adjusting the BPM value.
- void `edit_callback_notepad` ()
A callback function for handling an edit to the screen-set notepad.
- bool `timer_callback` ()
This function is the GTK timer callback, used to draw our current time and BPM on_events (the main window).
- void `learn_toggle` ()
Toggle the group-learn status.
- void `open_performance_edit` ()
Opens the Performance Editor (Song Editor).
- void `sequence_key` (int seq)
Use the sequence key to toggle the playing of an active pattern in the current screen-set.
- void `update_window_title` ()
Updates the title shown in the title bar of the window.
- void `toLower` (std::string &)
Converts a string to lower-case letters.
- void `file_new` ()
A callback function for the File / New menu entry.
- void `file_open` ()
A callback function for the File / Open menu entry.
- void `file_save` ()
A callback function for the File / Save menu entry.
- void `file_save_as` ()
A callback function for the File / Save As menu entry.
- void `file_exit` ()
A callback function for the File / Exit menu entry.
- void `new_file` ()
Actually does the work of setting up for a new file.
- bool `save_file` ()
Saves the current state in a MIDI file.
- void `choose_file` ()
Creates a file-chooser dialog.
- int `query_save_changes` ()
Queries the user to save the changes made while the application was running.

- bool [is_save](#) ()
If the data is modified, then the user is queried, and the file is save if okayed.
- bool [install_signal_handlers](#) ()
Installs the signal handlers and pipe code.
- bool [signal_action](#) (Glib::IOCondition condition)
Handles saving or exiting actions when signalled.
- bool [on_delete_event](#) (GdkEventAny *a_e)
This callback function handles a delete event from ...?
- bool [on_key_press_event](#) (GdkEventKey *a_ev)
Handles a key press event.
- bool [on_key_release_event](#) (GdkEventKey *a_ev)
Handles a key release event.
- virtual void [on_grouplearnchange](#) (bool state)
Notification handler for learn mode toggle.

Static Private Member Functions

- static void [handle_signal](#) (int sig)
This function is the handler for system signals (SIGUSR1, SIGINT...) It writes a message to the pipe and leaves as soon as possible.

Private Attributes

- Gtk::MenuBar * [m_menubar](#)
Theses objects support the menu and its sub-menus.
- [mainwid](#) * [m_main_wid](#)
The biggest sub-components of mainwnd.
- [maintime](#) * [m_main_time](#)
Is this the bar at the top that shows moving squares?
- [perfedit](#) * [m_perf_edit](#)
A pointer to the song/performance editor.
- [options](#) * [m_options](#)
A pointer to the program options.
- Gdk::Cursor [m_main_cursor](#)
Mouse cursor?
- Gtk::Button * [m_button_learn](#)
This button is the learn button, otherwise known as the "L" button.
- Gtk::Button * [m_button_stop](#)
Implements the red square stop button.
- Gtk::Button * [m_button_play](#)
Implements the green triangle play button.
- Gtk::Button * [m_button_perfedit](#)
The button for bringing up the Song Editor (Performance Editor).
- Gtk::SpinButton * [m_spinbutton_bpm](#)
The spin/adjustment controls for the BPM (beats-per-minute) value.
- Gtk::SpinButton * [m_spinbutton_ss](#)
The spin/adjustment controls for the screen set value.
- Gtk::SpinButton * [m_spinbutton_load_offset](#)
The spin/adjustment controls for the load offset value.
- Gtk::Entry * [m_entry_notes](#)

What is this?

- sigc::connection [m_timeout_connect](#)

Provides a timeout handler.

- int [m_ppqn](#)

Saves the PPQN value obtained from the MIDI file (or the default value, `c_ppqn`, if `SEQ64_USE_DEFAULT_PPQN` was specified in reading the MIDI file.

Static Private Attributes

- static int [m_sigpipe](#) [2]

Interesting; what is this used for.

Additional Inherited Members

7.23.1 Constructor & Destructor Documentation

7.23.1.1 seq64::mainwnd::mainwnd (perform & p)

This constructor is way too large; it would be nicer to provide a number of well-named initialization functions.

Parameters

<i>p</i>	Refers to the main performance object.
----------	--

Todo Offload most of the work into an initialization function like options does; make the perform parameter a reference; valgrind flags `m_tooltips` as lost data, but if we try to manage it ourselves, many more leaks occur.

File menu items, their accelerator keys, and their hot keys.

View menu items and their hot keys.

Help menu items

Top panel items, including the logo (updated for the new version of this application) and the "timeline" progress bar.

7.23.2 Member Function Documentation

7.23.2.1 void seq64::mainwnd::open_file (const std::string & fn)

We leave the `ppqn` parameter set to the `SEQ64_USE_DEFAULT` for now, to preserve the legacy behavior of using `c_ppqn`, and scaling the running time against the PPQN read from the MIDI file. Later, we can provide a value like 0, that will certainly be changed by reading the MIDI file.

We don't need to specify the "propformat" parameter of the `midifile` constructor when reading the MIDI file, since reading handles both the old and new formats.

Parameters

<i>fn</i>	Provides the file-name for the MIDI file to be opened.
-----------	--

7.23.2.2 void seq64::mainwnd::ppqn (int ppqn) [inline]

```
m_ppqn = (ppqn == SEQ64_USE_DEFAULT_PPQN) ? global_ppqn : ppqn ;
```

7.23.2.3 void seq64::mainwnd::file_import_dialog () [private]

Note that every track of the MIDI file will be imported, even if the track is only a label track (without any MIDI events), or a very long track.

The main difference between the Open operation and the Import operation seems to be that the latter can read MIDI files into a screen-set greater than screen-set 0. No, that's not true, so far. No matter what the current screen-set setting, the import is appended after the current data in screen-set 0. Then, if it overflows that screen-set, the overflow goes into the next screen-set.

It might be nice to have the option of importing a MIDI file into a specific screen-set, for better organization. Set versus append.

Todo We need to look into the Import process and document it better.

7.23.2.4 void seq64::mainwnd::about_dialog () [private]

I (Chris) took the liberty of tacking my name at the end, and hope to eventually have done enough work to warrant having it there.

7.23.2.5 void seq64::mainwnd::adj_callback_ss () [private]

Sets the screen-set value in the Performance/Song window, the Patterns, and something about setting the text based on a screen-set notepad from the Performance/Song window.

Screen-set notepad?

7.23.2.6 bool seq64::mainwnd::timer_callback () [private]

Note

When Sequencer64 first starts up, and no MIDI tune is loaded, the call to [mainwid::update_markers\(\)](#) leads to trying to do some work on sequences that don't yet exist.

7.23.2.7 void seq64::mainwnd::open_performance_edit () [private]

Todo Try to find a way to set m_modified only if the song editor actually changes something, instead of just because it was opened.

7.23.2.8 void seq64::mainwnd::update_window_title () [private]

Note that the name of the application is obtained by the "(SEQ64_PACKAGE)" construction.

The format of the caption bar is the name of the package/application, followed by the file-specification (shortened if necessary so that the name of the file itself can be seen), ending with the PPQN value in parentheses.

7.23.2.9 bool seq64::mainwnd::save_file () [private]

Here we specify the current value of m_ppqn, which was set when reading the MIDI file.

7.23.2.10 `bool seq64::mainwnd::signal_action (Glib::IOCondition condition) [private]`

Returns

Returns true if the signalling was able to be completed, even if it was an unexpected signal.

7.23.2.11 `bool seq64::mainwnd::on_delete_event (GdkEventAny * a_e) [private]`

Any changed data is saved. If the pattern is playing, then it is stopped.

7.23.2.12 `bool seq64::mainwnd::on_key_press_event (GdkEventKey * a_ev) [private]`

It also handles the control-key and modifier-key combinations matching the entries in its list of if statements.

Todo Test this functionality in old and new application.

7.23.2.13 `bool seq64::mainwnd::on_key_release_event (GdkEventKey * a_ev) [private]`

Is this worth turning into a switch statement? Or offloading to a perform member function? The latter.

Todo Test this functionality in old and new application.

Returns

Always returns false.

7.23.2.14 `void seq64::mainwnd::on_grouplearnchange (bool state) [private],[virtual]`

This handler responds to a learn-mode change from [perf\(\)](#).

Reimplemented from [seq64::performcallback](#).

7.23.3 Field Documentation

7.23.3.1 `int seq64::mainwnd::m_sigpipe [static],[private]`

This static member provides a couple of pipes for signalling/messaging.

7.23.3.2 `mainwid* seq64::mainwnd::m_main_wid [private]`

The first is the Patterns Panel.

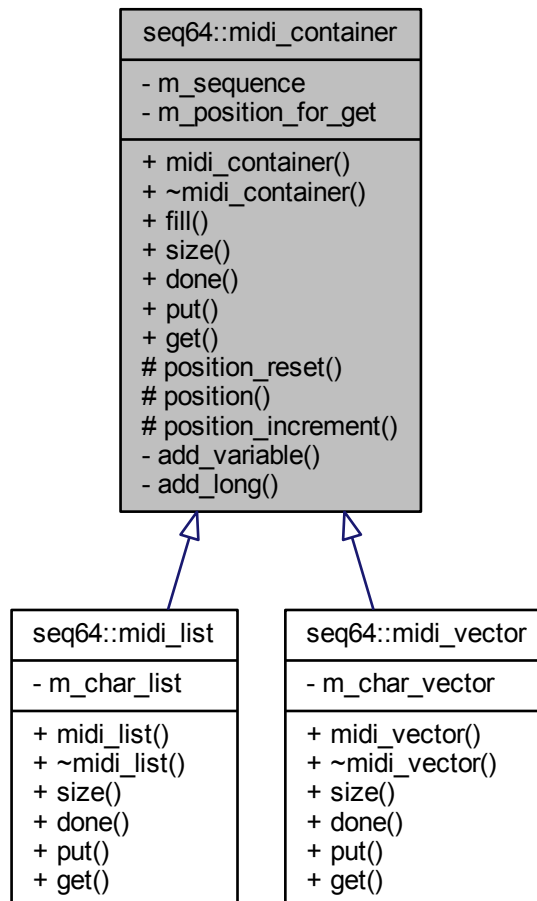
7.23.3.3 `Gtk::SpinButton* seq64::mainwnd::m_spinbutton_load_offset [private]`

However, where is this button located? It is handled in the code, but I've never seen the button!

7.24 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.

Private Member Functions

- void [add_variable](#) (long v)
This function masks off the lower 8 bits of the long parameter, then shifts it right 7, and, if there are still set bits, it encodes it into the buffer in reverse order.
- void [add_long](#) (long x)
What is the difference between this function and [add_list_var\(\)](#)?

Private Attributes

- [sequence](#) & [m_sequence](#)
Provide a hook into a sequence so that we can exchange data with a sequence object.
- unsigned int [m_position_for_get](#)
Provides the position in the container when making a series of [get\(\)](#) calls on the container.

7.24.1 Member Function Documentation

7.24.1.1 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

<i>tracknumber</i>	Provides the track number. This number is masked into the track information.
--------------------	--

7.24.1.2 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](#).

7.24.1.3 virtual midibyte seq64::midi_container::get () [pure virtual]

It also increments [m_position_for_get](#).

Implemented in [seq64::midi_list](#), and [seq64::midi_vector](#).

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

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

7.24.1.5 `void seq64::midi_container::add_variable (long v)` `[private]`

This function "replaces" `sequence::add_list_var()`.

7.24.1.6 `void seq64::midi_container::add_long (long x)` `[private]`

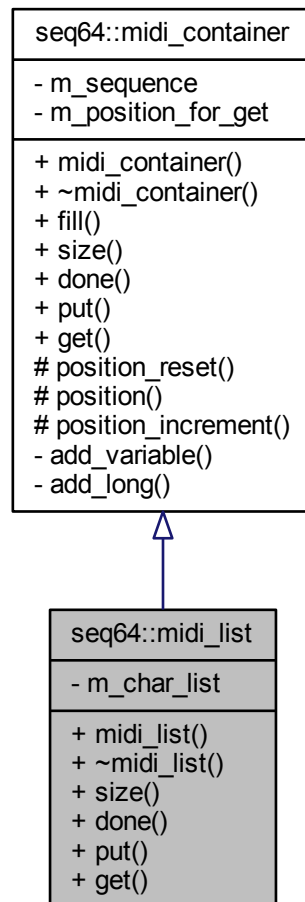
This function "replaces" `sequence::add_long_list()`.

This was a *global* internal function called `addLongList()`. Let's at least make it a private member now, and hew to the naming conventions of this class.

7.25 `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.

Private Types

- `typedef std::list< midibyte > CharList`

Provides the type of this container.

Private Attributes

- `CharList m_char_list`

The container itself.

Additional Inherited Members

7.25.1 Member Typedef Documentation

7.25.1.1 `typedef std::list<midibyte> seq64::midi_list::CharList` `[private]`

This type is basically the same as the container used in the midifile module, and almost identical to the CharList type defined in the sequence module.

7.25.2 Member Function Documentation

7.25.2.1 `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](#).

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

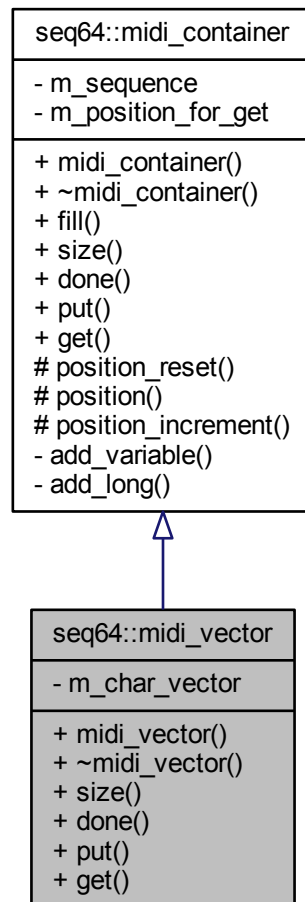
In this implement, m_position_for_get is not used. The elements of the container are popped of backward!

Implements [seq64::midi_container](#).

7.26 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.

Private Types

- `typedef std::vector< midibyte > CharVector`

Provides the type of this container.

Private Attributes

- `CharVector m_char_vector`

The container itself.

Additional Inherited Members

7.26.1 Member Function Documentation

7.26.1.1 `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](#).

7.26.1.2 `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](#).

7.27 seq64::midifile Class Reference

This class handles the parsing and writing of MIDI files.

Public Member Functions

- `midifile (const std::string &name, int ppqn=SEQ64_USE_DEFAULT_PPQN, 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.
- `int ppqn () const`
'Getter' function for member m_ppqn Provides a way to get the actual value of PPQN used in processing the sequences when [parse\(\)](#) was called.

Private Member Functions

- `unsigned long parse_prop_header (int file_size)`
Parse the proprietary header, figuring out if it is the new format, or the legacy format, for sequencer-specific data.
- `bool parse_proprietary_track (perform &a_perf, int file_size)`
After all of the conventional MIDI tracks are read, we're now at the "proprietary" Seq24 data section, which describes the various features that Seq24 supports.

- unsigned long [read_long](#) ()
Reads 4 bytes of data using [read_byte\(\)](#).
- unsigned short [read_short](#) ()
Reads 2 bytes of data using [read_byte\(\)](#).
- unsigned char [read_byte](#) ()
Reads 1 byte of data directly into the `m_data` vector, incrementing `m_pos` after doing so.
- unsigned long [read_varinum](#) ()
Read a MIDI Variable-Length Value (VLV), which has a variable number of bytes.
- void [write_long](#) (unsigned long)
Writes 4 bytes, using the [write_byte\(\)](#) function.
- void [write_short](#) (unsigned short)
Writes 2 bytes, using the [write_byte\(\)](#) function.
- void [write_byte](#) (unsigned char c)
Writes 1 byte.
- void [write_varinum](#) (unsigned long)
Writes a MIDI Variable-Length Value (VLV), which has a variable number of bytes.
- void [write_track_name](#) (const std::string &trackname)
Writes out a track name.
- void [write_seq_number](#) (unsigned short seqnum)
Writes out a sequence number.
- void [write_track_end](#) ()
Writes out the end-of-track marker.
- void [write_prop_header](#) (unsigned long tag, long len)
We want to write:
- bool [write_proprietary_track](#) (perform &a_perf)
Writes out the proprietary section, using the new format if the legacy format is not in force.
- long [varinum_size](#) (long len) const
Calculates the length of a variable length value.
- long [prop_item_size](#) (long datalen) const
Calculates the size of a proprietary item, as written by the [write_prop_header\(\)](#) function, plus whatever is called to write the data.
- long [track_name_size](#) (const std::string &trackname) const
Calculates the size of a trackname and the meta event that specifies it.
- long [seq_number_size](#) () const
Returns the size of a sequence-number event, which is always 5 bytes, plus one byte for the delta time that precedes it.
- long [track_end_size](#) () const
Returns the size of a track-end event, which is always 3 bytes.

Private Attributes

- int [m_pos](#)
Holds the position in the MIDI file.
- const std::string [m_name](#)
The unchanging name of the MIDI file.
- std::vector< unsigned char > [m_data](#)
This vector of characters holds our MIDI data.
- std::list< unsigned char > [m_char_list](#)
Provides a list of characters.
- bool [m_new_format](#)
Use the new format for the proprietary footer section of the Seq24 MIDI file.
- int [m_ppqn](#)
Provides the current value of the PPQN, which used to be the constant `c_ppqn`.

7.27.1 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.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 `seq64::midifile::midifile (const std::string & name, int ppqn = SEQ64_USE_DEFAULT_PPQN, bool propformat = true)`

Parameters

<i>name</i>	Provides the name of the MIDI file to be read or written.
<i>ppqn</i>	Provides the initial value of the PPQN setting. It is handled differently for parsing (reading) versus writing the MIDI file. <ul style="list-style-type: none"> • Reading. <ul style="list-style-type: none"> – If set to SEQ64_USE_DEFAULT_PPQN, the legacy application behavior is used. The m_ppqn member is set to the default PPQN, global_ppqn = c_ppqn. The value read from the MIDI file, ppqn, is then use to scale the running-time of the sequence relative to global_ppqn. – Otherwise, m_ppqn is set to the value read from the MIDI file. No scaling is done. Since the value gets written, specify ppqn as 0, an obviously bogus value. • Writing. This value is written to the MIDI file in the header chunk of the song.
<i>propformat</i>	If true, write out the MIDI file using the new MIDI-compliant sequencer-specific format for 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.

7.27.3 Member Function Documentation

7.27.3.1 `bool seq64::midifile::parse (perform & a_perf, int screenset)`

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?

Parameters

<i>a_perf</i>	Provides a reference to the perform object into which sequences/tracks are to be added.
<i>screenset</i>	The screen-set to be updated.

Returns

Returns true if the parsing succeeded.

7.27.3.2 `bool seq64::midifile::write (perform & a_perf)`

Parameters

<i>a_perf</i>	Provides the object that will contain and manage the entire performance.
---------------	--

Returns

Returns true if the write operations succeeded.

Warning

This writing backwards reverses the order of some events that are otherwise equivalent in time-stamp and rank. But it must be done this way, otherwise many items are backward.

7.27.3.3 `int seq64::midifile::ppqn () const [inline]`

The PPQN will be either `c_ppqn` (legacy behavior) or the value read from the file, depending on the `ppqn` parameter passed to the `midifile` constructor.

7.27.3.4 `unsigned long seq64::midifile::parse_prop_header (int file_size) [private]`

The new format creates a final track chunk, starting with "MTrk". Then comes the delta-time (here, 0), and the event. An event is a MIDI event, a SysEx event, or a Meta event.

A MIDI Sequencer Specific meta message includes either a delta time or absolute time, and the MIDI Sequencer Specific event encoded as follows:

```
0xFF 0x7F 0x02 length data
```

For convenience, this function first checks the amount of file data left. Then it reads a long value. If the value starts with FF, then that signals the new format. Otherwise, it is probably the old format, and the long value is a control tag (0x242400nn), which can be returned immediately.

If it is the new format, we back up to the FF, then get the next byte, which should be a 7F. If so, then we read the length (a variable length value) of the data, and then read the long value, which should be the control tag, which, again, is returned by this function.

Note

Most sequencers seem to be tolerant of both the lack of an "MTrk" marker and of the presence of an unwrapped control tag, and so can handle both the old and new formats of the final proprietary track.

Parameters

<i>file_size</i>	The size of the data file. This value is compared against the member <code>m_pos</code> (the position inside <code>m_data[]</code>), to make sure there is enough data left to process.
------------------	--

Returns

Returns the control-tag value found. These are the values, such as `c_midich`, found in the `globals` module, that indicate the type of sequencer-specific data that comes next. If there is not enough data to process, then 0 is returned.

7.27.3.5 `bool seq64::midifile::parse_proprietary_track (perform & a_perf, int file_size) [private]`

It consists of series of tags:

- `c_midictrl`
- `c_midiclocks`
- `c_notes`
- `c_bpmtag`
- `c_mutegroups`

(There are more tags defined in the `globals` module, but they are not used in this function. This doesn't quite make sense, as there are also some "triggers" values, and we're pretty sure the application uses them.)

The format is (1) tag ID; (2) length of data; (3) the data.

Change Note ca 2015-08-16 First, we separate out this function for a little more clarify. Then we add code to handle reading both the legacy Seq24 format and the new, MIDI-compliant format. Note that the format is not quite correct, since it doesn't handle a MIDI manufacturer's ID, making it a single byte that is part of the data.

Parameters

<i>a_perf</i>	The performance object that is being set via the incoming MIDI file.
<i>file_size</i>	The file size as determined in the parse() function.

There is also an implicit parameter in the `m_pos` member variable.

7.27.3.6 `unsigned long seq64::midifile::read_long () [private]`

Warning

This code looks endian-dependent and integer-size dependent.

7.27.3.7 `unsigned short seq64::midifile::read_short () [private]`

Warning

This code looks endian-dependent.

7.27.3.8 `unsigned long seq64::midifile::read_varinum () [private]`

This function reads the bytes while bit 7 is set in each byte. Bit 7 is a continuation bit. See [write_varinum\(\)](#) for more information.

7.27.3.9 `void seq64::midifile::write_long (unsigned long a_x) [private]`

Warning

This code looks endian-dependent.

7.27.3.10 `void seq64::midifile::write_short (unsigned short a_x) [private]`

Warning

This code looks endian-dependent.

7.27.3.11 `void seq64::midifile::write_byte (unsigned char c) [inline],[private]`

The byte is written to the `m_char_list` member, using a call to `push_back()`.

7.27.3.12 `void seq64::midifile::write_varinum (unsigned long value) [private]`

A MIDI file Variable Length Value is stored in bytes. Each byte has two parts: 7 bits of data and 1 continuation bit. The highest-order bit is set to 1 if there is another byte of the number to follow. The highest-order bit is set to 0 if this byte is the last byte in the VLV.

To recreate a number represented by a VLV, first you remove the continuation bit and then concatenate the leftover bits into a single number.

To generate a VLV from a given number, break the number up into 7 bit units and then apply the correct continuation bit to each byte.

In theory, you could have a very long VLV number which was quite large; however, in the standard MIDI file specification, the maximum length of a VLV value is 5 bytes, and the number it represents can not be larger than 4 bytes.

Here are some common cases:

- Numbers between 0 and 127 (0x7F) are represented by a single byte.
- 0x80 is represented as "0x81 0x00".
- 0xFFFFFFFF (the largest number) is represented as "0xFF 0xFF 0xFF 0x7F".

Also see the [varinum_size\(\)](#) function.

7.27.3.13 `void seq64::midifile::write_track_name (const std::string & trackname) [private]`

Note that we have to precede this "event" with a delta time value, set to 0.

7.27.3.14 `void seq64::midifile::write_seq_number (unsigned short seqnum) [private]`

The format is "FF 00 02 ss ss", where "02" is actually the constant length of the data. We have to precede these values with a 0 delta time, of course.

Now, for sequence 0, an alternate format 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 assumed to increment. This application doesn't bother with that shortcut.

7.27.3.15 void seq64::midifile::write_prop_header (unsigned long *control_tag*, long *data_length*) [private]

- 0x4D54726B. The track tag "MTrk". The MIDI spec requires that software can skip over non-standard chunks. "Prop"? Would require a fix to midicvt.
- 0xaabbccdd. The length of the track. This needs to be calculated somehow.
- 0x00. A zero delta time.
- 0x7f7f, The sequence number, a special value, well out of our normal range.
- The name of the track:
 - "Seq24-Spec"
 - "Sequencer24-S"

Then follows the proprietary data, written in the normal manner.

Finally, tack on the track-end meta-event.

Components of final track size:

```
-# Delta time. 1 byte, always 0x00.
-# Sequence number. 5 bytes. OPTIONAL. We won't write it.
-# Track name. 3 + 10 or 3 + 15
-# Series of proprietary specs:
  -# Prop header:
    -# If legacy format, 4 bytes.
    -# Otherwise, 2 bytes + varinum_size(length) + 4 bytes.
    -# Length of the prop data.
-# Track End. 3 bytes.
```

Writes a "proprietary" Seq24 footer header in either the new MIDI-compliant format, or the legacy Seq24 format. This function does not write the data. It replaces calls such as "write_long(c_midich)" in the proprietary section of [write\(\)](#).

The legacy format just writes the control tag (0x242400xx). The new format writes 0x00 0xFF 0x7F len 0x242400xx; the first 0x00 is the delta time.

In the new format, the 0x24 is a kind of "manufacturer ID". At <http://www.midi.org/techspecs/manid.php> we see that most manufacturer IDs start with 0x00, and are thus three bytes long, or start with codes at 0x40 and above. Similarly, <http://sequence15.blogspot.com/2008/12/midi-manufacturer-ids.html> shows that no manufacturer uses 0x24.

Warning

Currently, the manufacturer ID is not handled; it is part of the data, which can be misleading in programs that analyze MIDI files.

Parameters

<i>control_tag</i>	Determines the type of sequencer-specific section to be written. It should be one of the value in the globals module, such as <code>c_midibus</code> or <code>c_mutegroups</code> .
<i>data_length</i>	The amount of data that will be written. This parameter does not count the length of the header itself.

7.27.3.16 bool seq64::midifile::write_proprietary_track (perform & *a_perf*) [private]

The first thing to do, for the new format only, is calculate the length of this big section of data. This was quite tricky; we tweaked and adjusted until the midicvt program handled the whole new-format file without emitting any errors.

7.27.3.17 `long seq64::midifile::varinum_size (long len) const` `[private]`

This function is needed when calculating the length of a track. Note that it handles only the following situations:

https://en.wikipedia.org/wiki/Variable-length_quantity

```
1 byte:  0x00 to 0x7F
2 bytes: 0x80 to 0x3FFF
3 bytes: 0x4000 to 0x001FFFFF
4 bytes: 0x200000 to 0x0FFFFFFF
```

Returns

Returns values as noted above. Anything beyond that range returns 0.

7.27.3.18 `long seq64::midifile::prop_item_size (long data_length) const` `[private]`

If using the new format, the length includes the sum of sequencer-specific tag (0xFF 0x7F) and the size of the variable-length value. Then, for legacy and new format, 4 bytes are added for the Seq24 MIDI control value, and the the data length is added.

7.27.3.19 `long seq64::midifile::seq_number_size () const` `[inline], [private]`**7.27.4 Field Documentation**7.27.4.1 `int seq64::midifile::m_pos` `[private]`

This is at least a 31-bit value in the recent architectures running Linux and Windows, so it will handle up to 2 Gb of data. This member is used as the offset into the `m_data` vector.

7.27.4.2 `std::vector<unsigned char> seq64::midifile::m_data` `[private]`

We could also use a string of characters, unsigned. This member is resized to the putative size of the MIDI file, in the `parse()` function. Then the whole file is read into it, as if it were an array. This member is an input buffer.

7.27.4.3 `std::list<unsigned char> seq64::midifile::m_char_list` `[private]`

The class pushes each MIDI byte into this list using the `write_byte()` function. Also note that the `write()` function calls `sequence::fill_list()` to fill a temporary `std::list<char>` (!) buffer, then writes that data *backwards* to this member. This member is an output buffer.

7.27.4.4 `bool seq64::midifile::m_new_format` `[private]`

In this new format, each sequencer-specific value (0x242400xx, as defined in the `globals` module) is preceded by the sequencer-specific prefix, 0xFF 0x7F len id/date). By default, this value is true, but the user can specify the `-legacy` (-l) option, or make a soft link to the `sequence24` binary called "seq24", to write the data in the old format. [We will eventually add the `-legacy` option to the `~/ .seq24rc` configuration file.] Note that reading can handle either format transparently.

7.28 seq64::options Class Reference

This class supports a full tabbed options dialog.

Inherits Dialog.

Private Types

- enum [button](#)
Defines buttons indices or IDs for some controls related to JACK.

Private Attributes

- [perform](#) & [m_mainperf](#)
The performance object to which some of these options apply.
- Gtk::Button * [m_button_ok](#)
The famous "OK" button's pointer.
- Gtk::Notebook * [m_notebook](#)
Not sure yet what this notebook is for.

7.28.1 Field Documentation

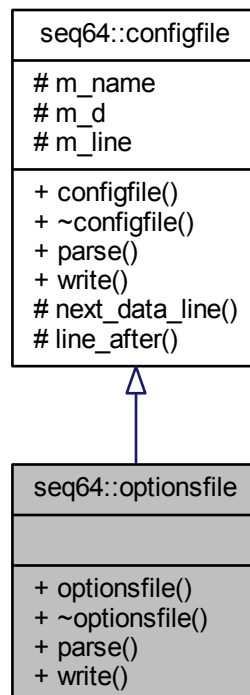
7.28.1.1 Gtk::Notebook* seq64::options::m_notebook [private]

Must be a GTK thang.

7.29 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

7.29.1 Detailed Description

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

7.29.2 Member Function Documentation

7.29.2.1 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 additional 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 specified:

- 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](#).

7.29.2.2 `bool seq64::optionsfile::write (const perform & a_perf) [virtual]`

Parameters

<i>a_perf</i>	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

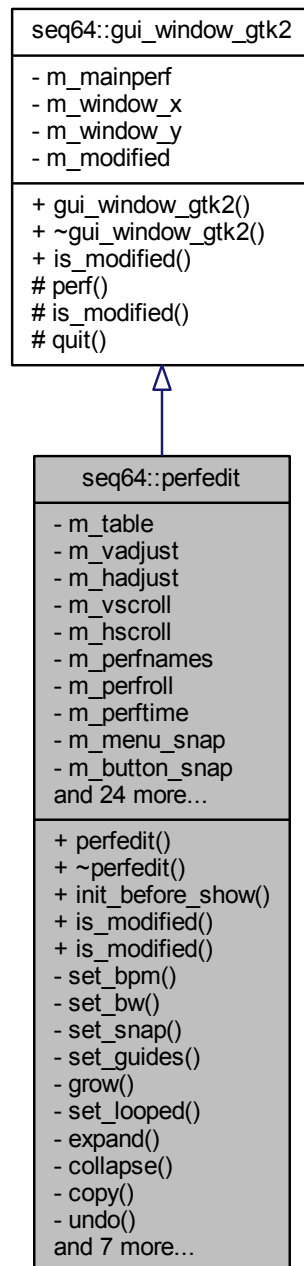
Returns true if the write operations all succeeded.

Implements [seq64::configfile](#).

7.30 seq64::perfedit Class Reference

This class supports a Performance Editor that is used to arrange the patterns/sequences defined in the patterns panel, I think.

Inheritance diagram for seq64::perfedit:



Public Member Functions

- [perfedit](#) ([perform](#) &p, int ppqn=SEQ64_USE_DEFAULT_PPQN, int bpm=DEFAULT_BEATS_PER_MEASURE, int bw=DEFAULT_BEAT_WIDTH)

Principal constructor, has a reference to a perform object.

- [~perfedit](#) ()

This rote constructor does nothing.

- void `init_before_show ()`
This function forwards its call to the `perfull` function of the same name.
- void `is_modified (bool flag)`
'Setter' function for member `m_modified`
- bool `is_modified () const`
'Getter' function for member `m_modified`

Private Member Functions

- void `set_bpm (int beats_per_measure)`
Sets the BPM (beats per minute) text and values to the given value, and then calls `set_guides()`.
- void `set_bw (int beat_width)`
Sets the BW (beat width, or the denominator in the time signature) text and values to the given value, and then calls `set_guides()`.
- void `set_snap (int snap)`
Sets the snap text and values to the given value, and then calls `set_guides()`.
- void `set_guides ()`
Sets the guides, which are the L and R user-interface elements.
- void `grow ()`
Increments the size of the `perfull` and `perftime` objects.
- void `set_looped ()`
Set the looping in the `perform` object.
- void `expand ()`
Implement the expand action.
- void `collapse ()`
Implement the collapse action.
- void `copy ()`
Implement the copy (actually, expand-and-copy) action.
- void `undo ()`
Implement the undo feature (Ctrl-Z).
- void `popup_menu (Gtk::Menu *menu)`
Opens the given popup menu.
- bool `timeout ()`
Handles a drawing timeout.
- void `start_playing ()`
Implement the playing.
- void `stop_playing ()`
Stop the playing.
- void `on_realize ()`
This callback function calls the base-class `on_realize()` function, and then connects the `perfed::timeout()` function to the Glib signal-timeout, with a redraw timeout of `m_redraw_ms`.
- bool `on_key_press_event (GdkEventKey *ev)`
This function is the callback for a key-press event.
- bool `on_delete_event (GdkEventAny *)`
All this callback function does is return false.

Private Attributes

- Gtk::Menu * `m_menu_bpm`
Menus for time signature, beats per measure, beat width.
- int `m_snap`
Set snap-to in "pulses".

Additional Inherited Members

7.30.1 Detailed Description

It has a seqroll and piano roll? No, it has a perform, a perfnames, a perfroll, and a perftime.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 `seq64::perfedit::perfedit (perform & p, int ppqn = SEQ64_USE_DEFAULT_PPQN, int bpm = DEFAULT_BEATS_PER_MEASURE, int bw = DEFAULT_BEAT_WIDTH)`

We've reordered the pointer members and put them in the initializer list to make the constructor a bit cleaner.

Parameters

<i>p</i>	Refers to the main performance object.
----------	--

Todo Offload most of the work into an initialization function like options does; make the perform parameter a reference.

7.30.2.2 `seq64::perfedit::~~perfedit ()`

We're going to have to run the application through valgrind to make sure that nothing is left behind.

7.30.3 Member Function Documentation

7.30.3.1 `void seq64::perfedit::init_before_show ()`

It does not seem to need to also forward to the perftime function of the same name.

7.30.3.2 `void seq64::perfedit::set_guides () [private]`

See the [set_snap\(\)](#) function.

7.30.3.3 `void seq64::perfedit::expand () [private]`

This action opens up a space events between the L and R (left and right) markers. This action is preceded by pushing an Undo operation in the perform object, moving its triggers, and telling the perfroll to redraw.

7.30.3.4 `void seq64::perfedit::collapse () [private]`

This action removes all events between the L and R (left and right) markers. This action is preceded by pushing an Undo operation in the perform object, not moving its triggers (they go away), and telling the perfroll to redraw.

7.30.3.5 `void seq64::perfedit::copy () [private]`

This action opens up a space events between the L and R (left and right) markers, and copies the information from the same amount of event that follow the R marker. This action is preceded by pushing an Undo operation in the perform object, copying its triggers, and telling the perfroll to redraw.

7.30.3.6 `void seq64::perfedit::undo () [private]`

We pop an Undo trigger, and then ask the perfrull to queue up a (re)drawing action.

7.30.3.7 `bool seq64::perfedit::timeout () [private]`

It redraws "dirty" sequences in the perfrull and the perfnames objects, and shows draw progress on the perfrull.

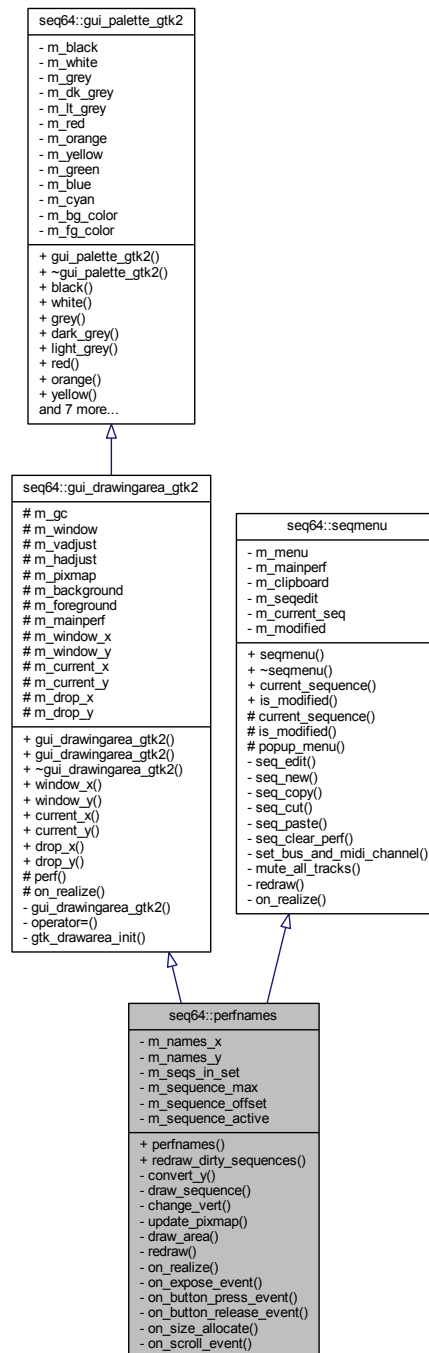
7.30.3.8 `void seq64::perfedit::start_playing () [inline],[private]`

JACK will be used if it is present and, in the application, enabled.

7.31 seq64::perfnames Class Reference

This class implements the left-side keyboard in the patterns window.

Inheritance diagram for seq64::perfnames:



Public Member Functions

- [perfnames](#) ([perform](#) &p, Gtk::Adjustment &vadjust)

Principal constructor for this user-interface object.

- void [redraw_dirty_sequences](#) ()

Redraws sequences that have been modified.

Private Member Functions

- int `convert_y` (int `y`)
Converts a y-value into a sequence number and returns it.
- void `draw_sequence` (int `sequence`)
Draw the given sequence.
- void `change_vert` ()
Change the vertical offset of a sequence/pattern.
- void `update_pixmap` ()
This function does nothing.
- void `draw_area` ()
This function does nothing.
- void `redraw` (int `sequence`)
Redraw the given sequence.
- void `on_realize` ()
Handles the callback when the window is realized.
- bool `on_expose_event` (GdkEventExpose *`ev`)
Handles an on-expose event.
- bool `on_button_press_event` (GdkEventButton *`ev`)
Provides the callback for a button press, and it handles only a left mouse button.
- bool `on_button_release_event` (GdkEventButton *`ev`)
Handles a button-release for the right button, bringing up a popup menu.
- void `on_size_allocate` (Gtk::Allocation &)
Handles a size-allocation event.
- bool `on_scroll_event` (GdkEventScroll *`ev`)
Handle the scrolling of the window.

Additional Inherited Members

7.31.1 Detailed Description

Obsolete Note the usage of virtual base classes. Since these can add some extra overhead, we should determine if we can do without the virtuality (and indeed it doesn't seem to be needed).

7.31.2 Constructor & Destructor Documentation

7.31.2.1 `seq64::perfnames::perfnames (perform & p, Gtk::Adjustment & vadjust)`

Weird is that the window (x,y) are set to (c_names_x, 100), when c_names_y is 22 in globals.h.

7.31.3 Member Function Documentation

7.31.3.1 `void seq64::perfnames::on_realize () [private]`

It first calls the base-class version of `on_realize()`. Then it allocates any additional resources needed.

7.31.3.2 `bool seq64::perfnames::on_expose_event (GdkEventExpose * a_e) [private]`

It draws all of the sequences.

7.31.3.3 void seq64::perfnames::on_size_allocate (Gtk::Allocation & a) [private]

It first calls the base-class version of this function.

7.32 seq64::perform Class Reference

This class supports the performance mode.

Public Member Functions

- [perform](#) (gui_assistant &mygui, int ppqn=SEQ64_USE_DEFAULT_PPQN)
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.
- int [sequence_count](#) () const
'Getter' function for member m_sequence_count It is better to call this getter before bothering to even try to use a sequence.
- int [sequence_max](#) () const
'Getter' function for member m_sequence_max
- 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 *seq, int perf)
Adds a pattern/sequence pointer to the list of patterns.

- void [delete_sequence](#) (int seq)
Deletes a pattern/sequence by number.
- bool [is_sequence_in_edit](#) (int seq)
Check if the pattern/sequence, given by number, has an edit in progress.
- void [clear_sequence_triggers](#) (int seq)
Clears the patterns/sequence for the given sequence, if it is active.
- long [get_tick](#) () const
'Getter' function for member m_tick
- void [set_left_tick](#) (long 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 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 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 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.
- midi_control * [get_midi_control_toggle](#) (unsigned int seq)
Retrieves a value from m_midi_cc_toggle[].
- midi_control * [get_midi_control_on](#) (unsigned int seq)
Retrieves a value from m_midi_cc_on[].
- midi_control * [get_midi_control_off](#) (unsigned int seq)
Retrieves a value from m_midi_cc_off[].
- void [handle_midi_control](#) (int control, bool state)
Handle the MIDI Control values that provide some automation for the application.
- const std::string & [get_screen_set_notepad](#) (int 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 ¬e)
Copies the given string into m_screen_set_notepad[].
- void [set_current_screen_set_notepad](#) (const std::string ¬e)
Sets the notepad text for the current screen-set.
- void [set_screenset](#) (int 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 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 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 group)

Will need to study this one more closely.
- void [start](#) (bool 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 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 seq, bool active)

Sets or unsets the active state of the given pattern/sequence number.
- void [set_was_active](#) (int seq)

Sets was-active flags: main, edit, perf, and names.
- bool [is_active](#) (int seq)

Checks the pattern/sequence for activity.
- bool [is_dirty_main](#) (int seq)

Checks the pattern/sequence for main-dirtiness.
- bool [is_dirty_edit](#) (int seq)

Checks the pattern/sequence for edit-dirtiness.
- bool [is_dirty_perf](#) (int seq)

Checks the pattern/sequence for perf-dirtiness.
- bool [is_dirty_names](#) (int seq)

Checks the pattern/sequence for names-dirtiness.
- void [new_sequence](#) (int seq)

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

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 tick)
Plays all notes to the current tick.
- void `set_orig_ticks` (long tick)
For every pattern/sequence that is active, sets the "original ticks" value for the pattern.
- void `set_bpm` (int 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 looping)
'Setter' function for member m_looping
- void `set_sequence_control_status` (int status)
If the given status is present in the c_status_snapshot, the playing state is saved.
- void `unset_sequence_control_status` (int status)
If the given status is present in the c_status_snapshot, the playing state is restored.
- void `set_group_mute_state` (int g_track, bool mute_state)
'Setter' function for member m_mute_group
- bool `get_group_mute_state` (int 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 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 [perftroll_key_event](#) (const [keystroke](#) &k, int drop_sequence)
Provided for [perftroll::on_key_press_event\(\)](#) and [perftroll::on_key_release_event\(\)](#) to call.

Private Member Functions

- bool [is_midi_control_valid](#) (unsigned int seq) const
Checks the parameter against c_midi_controls.
- bool [is_screenset_valid](#) (int screenset) const
Checks the screenset against c_max_sets.
- void [set_running](#) (bool running)
'Setter' function for member m_running
- void [set_playback_mode](#) (bool playbackmode)
'Setter' function for member m_playback_mode
- bool [is_seq_valid](#) (int seq) const
Provides common code to check for the bounds of a sequence number.
- bool [is_mseq_valid](#) (int seq) const
Validates the sequence number, which is important since they're currently used as array indices.
- void [install_sequence](#) ([sequence](#) *seq, int seqnum)
A private helper function for [add_sequence\(\)](#).
- void [inner_start](#) (bool state)
Locks on m_condition_var.
- void [inner_stop](#) ()
Unconditionally, and without locking, clears the running status, resets the sequences, and set m_usemidiclock false.
- 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.
- int [clamp_track](#) (int track) const
Provides common code to keep the track value valid.

Private Attributes

- [gui_assistant](#) & [m_gui_support](#)
Support for a wide range of GUI-related operations.
- bool [m_mute_group](#) [c_gmute_tracks]
Mute group support.
- int [m_playing_screen](#)
Playing screen support.
- [sequence](#) * [m_seqs](#) [c_max_sequence]
Provides a vector of patterns/sequences.
- mastermidibus [m_master_bus](#)
Provides our MIDI buss.
- pthread_t [m_out_thread](#)

Provides information for managing pthreads.

- bool [m_playback_mode](#)

Specifies the playback mode.

- long [m_tick](#)

MIDI Clock support.

Friends

- int [jack_sync_callback](#) (jack_transport_state_t state, jack_position_t *pos, void *arg)

Global functions for JACK support and JACK sessions.

7.32.1 Detailed Description

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

7.32.2 Constructor & Destructor Documentation

7.32.2.1 `seq64::perform::perform (gui_assistant & mygui, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

Parameters

<i>mygui</i>	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 <code>m_gui_support</code> member using the gui() accessor function.
--------------	--

7.32.2.2 `seq64::perform::~~perform ()`

Finally, any active patterns/sequences are deleted.

7.32.3 Member Function Documentation

7.32.3.1 `int seq64::perform::sequence_count () const [inline]`

In many cases at startup, or when loading a file, there are no sequences yet, and still the code calls functions that try to access them.

7.32.3.2 `void seq64::perform::init ()`

Who calls this routine? The `main()` routine of the application.

7.32.3.3 `void seq64::perform::clear_all ()`

The `mainwnd` module calls this function.

7.32.3.4 `void seq64::perform::launch_input_thread ()`

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

7.32.3.5 void seq64::perform::launch_output_thread ()

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

7.32.3.6 void seq64::perform::init_jack ()

Who calls this routine? The main() routine of the application, and the options module.

7.32.3.7 void seq64::perform::add_sequence (sequence * seq, int prefnum)

No check is made for a null pointer.

This function checks for the preferred sequence number. This is the number that was specified by the Sequence Number meta-event for the current track. If the preferred sequence number is in the valid range (0 to m_sequence_max) and it is not active, add it and activate it.

Otherwise, iterate through all patterns from prefnum 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 *prefnum* 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

<i>seq</i>	The pointer to the pattern/sequence to add.
<i>prefnum</i>	The preferred sequence number of the pattern, as explained above. If this value is out-of-range, then it is basically ignored.

7.32.3.8 void seq64::perform::delete_sequence (int seq)

We now also solidify the deletion by setting the pointer to null after deletion.

7.32.3.9 void seq64::perform::clear_sequence_triggers (int seq)

Parameters

<i>seq</i>	Provides the desired sequence. Hopefull, the is_active() function validates this value.
------------	---

7.32.3.10 void seq64::perform::move_triggers (bool a_direction)

Parameters

<i>a_direction</i>	Specifies the desired direction; false = left, true = right.
--------------------	--

7.32.3.11 void seq64::perform::copy_triggers ()

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

7.32.3.12 midi_control * seq64::perform::get_midi_control_toggle (unsigned int seq)

Parameters

<i>seq</i>	Provides a control value (such as <code>c_midi_control_bpm_up</code>) to use to retrieve the desired <code>midi_control</code> object. Note that this value is unsigned simply to make the legality check of the parameter easier.
------------	---

7.32.3.13 `midi_control * seq64::perform::get_midi_control_on (unsigned int seq)`

Parameters

<i>seq</i>	Provides a control value (such as <code>c_midi_control_bpm_up</code>) to use to retrieve the desired <code>midi_control</code> object.
------------	---

7.32.3.14 `midi_control * seq64::perform::get_midi_control_off (unsigned int seq)`

Parameters

<i>seq</i>	Provides a control value (such as <code>c_midi_control_bpm_up</code>) to use to retrieve the desired <code>midi_control</code> object.
------------	---

7.32.3.15 `const std::string & seq64::perform::get_screen_set_notepad (int screenset) const`

Parameters

<i>screenset</i>	The ID number of the string set, an index into the <code>m_screen_set_notepad[]</code> 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.

7.32.3.16 `void seq64::perform::set_screen_set_notepad (int screenset, const std::string & notepad)`

Parameters

<i>screenset</i>	The ID number of the string set, an index into the <code>m_screen_set_XXX[]</code> arrays.
<i>notepad</i>	Provides the string data 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.

7.32.3.17 `void seq64::perform::set_screenset (int ss)`

Parameters

<i>ss</i>	The index of the desired string set. It is forced to range from 0 to <code>c_max_sets - 1</code> .
-----------	--

7.32.3.18 `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.

7.32.3.19 void seq64::perform::unset_mode_group_learn ()

Then unsets the group-learn mode flag..

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

Parameters

<i>a_group</i>	Provides the group to mute. Note that this parameter is essentially a track or sequence number.
----------------	---

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

Parameters

<i>a_state</i>	What does this state mean?
----------------	----------------------------

7.32.3.22 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`?

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

Warning

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

7.32.3.24 void seq64::perform::all_notes_off ()

Then flush the MIDI buss.

7.32.3.25 void seq64::perform::set_active (int *seq*, bool *active*)

Parameters

<i>seq</i>	Provides the prospective sequence number.
<i>active</i>	True if the sequence is to be set to the active state.

7.32.3.26 void seq64::perform::set_was_active (int *seq*)

Why do we need this routine?

Parameters

<i>seq</i>	The pattern number. It is checked for invalidity.
------------	---

7.32.3.27 bool seq64::perform::is_active (int *seq*)

Parameters

<i>seq</i>	The pattern number. It is checked for invalidity. This can lead to "too many" (i.e. redundant) checks.
------------	--

Returns

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

7.32.3.28 bool seq64::perform::is_dirty_main (int *seq*)**Parameters**

<i>seq</i>	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.

7.32.3.29 bool seq64::perform::is_dirty_edit (int *seq*)**Parameters**

<i>seq</i>	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.

7.32.3.30 bool seq64::perform::is_dirty_perf (int *seq*)**Parameters**

<i>seq</i>	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.

7.32.3.31 bool seq64::perform::is_dirty_names (int *seq*)**Parameters**

<i>seq</i>	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.

7.32.3.32 void seq64::perform::new_sequence (int *seq*)

Then it activates the pattern.

It doesn't deal with thrown exceptions.

7.32.3.33 `sequence * seq64::perform::get_sequence (int seq)`

Parameters

<i>seq</i>	The prospective sequence number. It is checked for validity. We cannot compare the sequence number versus the <code>sequence_count()</code> , because the current implementation can have inactive holes (null pointers) interspersed with active pointers.
------------	---

Returns

Returns the value of `m_seqs[seq]` if `seq` is valid. Otherwise, a null pointer is returned.

7.32.3.34 `void seq64::perform::reset_sequences ()`

Then flush the MIDI buss.

7.32.3.35 `void seq64::perform::play (long 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

<i>tick</i>	Provides the tick at which to start playing.
-------------	--

7.32.3.36 `void seq64::perform::set_orig_ticks (long tick)`

Parameters

<i>tick</i>	
-------------	--

7.32.3.37 `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.

7.32.3.38 `void seq64::perform::set_sequence_control_status (int a_status)`

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

7.32.3.39 `void seq64::perform::unset_sequence_control_status (int a_status)`

Then the given status is reversed in `m_control_status`.

7.32.3.40 `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.

7.32.3.41 `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?

7.32.3.42 `void seq64::perform::set_offset (int offset) [inline]`

Sets `m_offset = offset * c_mainwnd_rows * c_mainwnd_cols;`

Parameters

<i>offset</i>	The desired offset.
---------------	---------------------

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

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

7.32.3.44 `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.

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

Actually does a lot of work in those function calls.

7.32.3.46 `int seq64::perform::increment_bpm () [inline]`

Actually does a lot of work in those function calls.

7.32.3.47 `void seq64::perform::set_input_bus (int bus, bool input_active)`

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

7.32.3.48 `bool seq64::perform::mainwnd_key_event (const keystroke & k)`

Returns

Returns true if the key was handled.

7.32.3.49 `bool seq64::perform::perfroll_key_event (const keystroke & k, int drop_sequence)`

Returns

Returns true if the key was handled.

7.32.3.50 `bool seq64::perform::is_midi_control_valid (unsigned int seq) const [inline], [private]`

Parameters

<i>seq</i>	The value that should be in the c_midi_xxx range.
------------	---

Returns

Returns true if the parameter is valid. For this function, no error print-out is generated.

7.32.3.51 `bool seq64::perform::is_screenset_valid (int screenset) const [inline], [private]`

Parameters

<i>screenset</i>	The prospective screenset value.
------------------	----------------------------------

Returns

Returns true if the parameter is valid. For this function, no error print-out is generated.

7.32.3.52 `bool seq64::perform::is_seq_valid (int seq) const [private]`

Also see the function [is_mseq_valid\(\)](#), which also checks the pointer stored in the m_seq[] array.

We considered checking the *seq* param against [sequence_count\(\)](#), but this function is called while creating sequences that add to that count, so we continue checking against the "container" size.

Parameters

<i>seq</i>	The sequencer number, in interval [0, m_sequence_max).
------------	--

Returns

Returns true if the sequence number is valid.

7.32.3.53 `bool seq64::perform::is_mseq_valid (int seq) const [private]`

It also evaluates the m_seq[seq] pointer value.

7.32.3.54 `void seq64::perform::install_sequence (sequence * seq, int seqnum) [private]`

It is common code and using it prevents inconsistencies. It assumes values have already been checked.

Parameters

<i>seq</i>	The pointer to the pattern/sequence to add.
<i>seqnum</i>	The sequence number of the pattern to be added.

7.32.3.55 `void seq64::perform::inner_start (bool a_state) [private]`

Then, if not `is_running()`, the playback mode is set to the given state. If that state is true, call `off_sequences()`. Set the running status, and signal the condition. Then unlock.

7.32.3.56 `void seq64::perform::set_key_event (unsigned int keycode, long sequence_slot) [private]`

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

It first removes the given key-code from the regular and reverse slot-maps. Then it removes the sequence-slot from the regular and reverse slot-maps.

Finally, it adds the sequence-slot with a key value of key-code, and adds the key-code with a value of sequence-slot.

Why are we erasing four items instead of just two?

7.32.3.57 `void seq64::perform::set_key_group (unsigned int keycode, long group_slot) [private]`

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

Compare it to the `set_key_events()` function.

7.32.3.58 `int seq64::perform::clamp_track (int track) const [inline], [private]`

Note the bug we found, where we checked for `track > c_seqs_in_set`, but set it to `c_seqs_in_set - 1` in that case!

7.32.4 Friends And Related Function Documentation

7.32.4.1 `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

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

7.32.5 Field Documentation

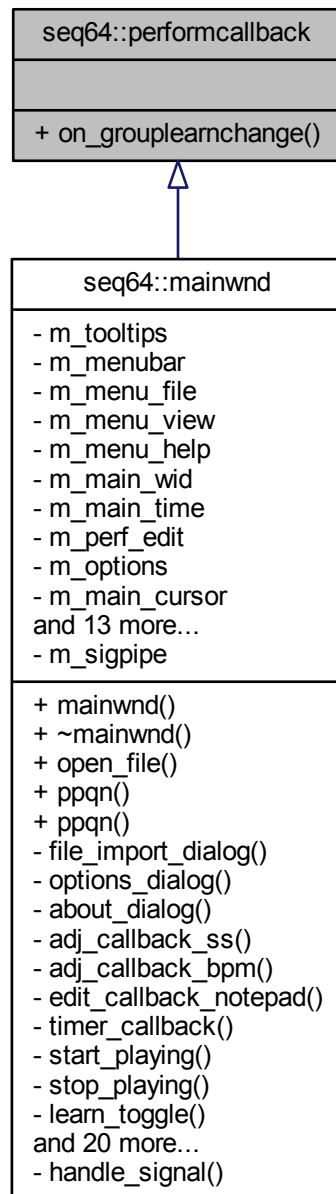
7.32.5.1 `bool seq64::perform::m_playback_mode [private]`

There are two, "live" and "song", but we're not yet sure what "true" indicates.

7.33 seq64::performcallback Struct Reference

Provides for notification of events.

Inheritance diagram for seq64::performcallback:



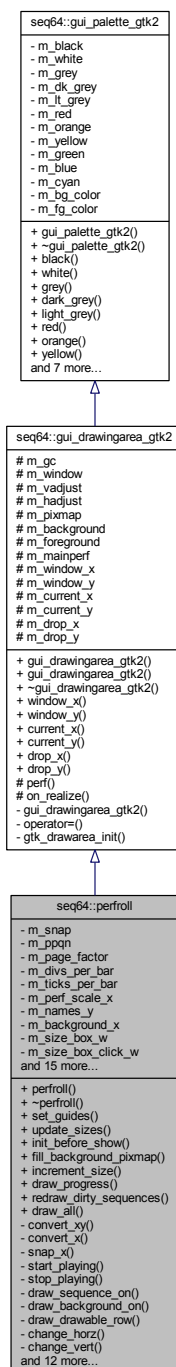
7.33.1 Detailed Description

Provide a response to a group-learn change event.

7.34 seq64::perfroll Class Reference

This class implements the performance roll user interface.

Inheritance diagram for seq64::perfroll:



Public Member Functions

- [perfroll](#) ([perform](#) &a_perf, Gtk::Adjustment &a_hadjust, Gtk::Adjustment &a_vadjust, int ppqn=SEQ64_US↵
E_DEFAULT_PPQN)
Principal constructor.
- [~perfroll](#) ()
This destructor deletes the interaction object.

- void [set_guides](#) (int a_snap, int a_measure, int a_beat)
This function sets the snap, measure, and beats members, fills in the background, and queues up a draw operation.
- void [update_sizes](#) ()
Updates the sizes of various items.
- void [init_before_show](#) ()
Sets the roll-lengths ticks member.
- void [fill_background_pixmap](#) ()
This function updates the background of the Performance roll.
- void [increment_size](#) ()
*Increments the value of m_roll_length_ticks by the PPQN * 512, then calls [update_sizes\(\)](#).*
- void [draw_progress](#) ()
Draws the progress line that shows where we are in the performance.
- void [redraw_dirty_sequences](#) ()
Redraws patterns/sequences that have been modified.
- void [draw_all](#) ()
Provides a very common sequence of calls used in perfull_input.

Private Member Functions

- void [convert_xy](#) (int x, int y, long &ticks, int &seq)
Converts a tick-offset....
- void [convert_x](#) (int x, long &ticks)
Converts a tick-offset on the x coordinate.
- void [snap_x](#) (int &x)
This function performs a 'snap' action on x.
- void [start_playing](#) ()
Start the performance playing.
- void [stop_playing](#) ()
Stop the performance playing.
- void [draw_sequence_on](#) (Glib::RefPtr< Gdk::Drawable > a_draw, int a_sequence)
Draws the given pattern/sequence on the given drawable area.
- void [draw_background_on](#) (Glib::RefPtr< Gdk::Drawable > a_draw, int a_sequence)
Draws the given pattern/sequence background on the given drawable area.
- void [draw_drawable_row](#) (Glib::RefPtr< Gdk::Drawable > a_dest, Glib::RefPtr< Gdk::Drawable > a_src, long a_y)
Not quite sure what this draws yet.
- void [change_horz](#) ()
Changes the 4-bar horizontal offset member and queues up a draw operation.
- void [change_vert](#) ()
Changes the 4-bar vertical offset member and queues up a draw operation.
- void [split_trigger](#) (int a_sequence, long a_tick)
Splits a trigger, whatever than means.
- void [on_realize](#) ()
Provides the on-realization callback.
- bool [on_expose_event](#) (GdkEventExpose *a_ev)
Handles the on-expose event.
- bool [on_button_press_event](#) (GdkEventButton *a_ev)
This callback function handles a button press by forwarding it to the interaction object's button-press function.
- bool [on_button_release_event](#) (GdkEventButton *a_ev)
This callback function handles a button release by forwarding it to the interaction object's button-release function.

- bool `on_motion_notify_event` (GdkEventMotion *a_ev)
Handles motion notification by forwarding it to the interaction object's motion-notification callback function.
- bool `on_scroll_event` (GdkEventScroll *a_ev)
Handles horizontal and vertical scrolling.
- bool `on_focus_in_event` (GdkEventFocus *)
This callback handles an in-focus event by setting the flag to HAS_FOCUS.
- bool `on_focus_out_event` (GdkEventFocus *)
This callback handles an out-of-focus event by resetting the flag HAS_FOCUS.
- void `on_size_allocate` (Gtk::Allocation &)
Upon a size allocation event, this callback calls the base-class version of this function, then sets m_window_x and m_window_y, and calls `update_sizes()`.
- bool `on_key_press_event` (GdkEventKey *a_p0)
This callback function handles a key-press event.
- void `on_size_request` (GtkRequisition *)
This callback throws away a size request.

Additional Inherited Members

7.34.1 Member Function Documentation

7.34.1.1 void seq64::perfroll::update_sizes ()

Note

Trying to figure out what the 16 is. So take the "bars-visible" calculation, the c_perf_scale_x value, assume that "ticks" is another name for "pulses", and assume that "beats" is a quarter note. Ignoring the numbers, the units come out to:

$$\text{bars} = \frac{\text{pixels} * \text{ticks} / \text{pixel}}{\text{ticks} / \text{beat} * \text{beats} / \text{bar}}$$

Thus, the 16 is a "beats per bar" or "beats per measure" value. This doesn't quite make sense, but there are 16 divisions per beat on the perfroll user-interface. So for now we'll call it the latter, and make a variable called "m_divs_per_bar", see its definition in the class initializer list.

7.34.1.2 void seq64::perfroll::init_before_show ()

First, it gets the largest trigger value among the active sequences. Then it truncates this value to the nearest PPQN * 16 ticks. Then it adds PPQN * 4096 ticks.

7.34.1.3 void seq64::perfroll::convert_xy (int x, int y, long & a_tick, int & a_seq) [private]

The results are returned via the a_tick and a_seq parameters.

7.34.1.4 void seq64::perfroll::convert_x (int x, long & tick) [private]

The result is returned via the a_tick parameter.

7.34.1.5 `void seq64::perfroll::snap_x(int & x) [private]`

- `m_snap` = number pulses to snap to
- `m_perf_scale_x` = number of pulses per pixel

Therefore `mod = m_snap/m_perf_scale_x` equals the number pixels to snap to.

7.34.1.6 `void seq64::perfroll::start_playing() [private]`

We need to keep in sync with `perfdit's start_playing()`... wish we could call it directly. Well, now we go to the source, calling `perform::start_playing()`.

7.34.1.7 `void seq64::perfroll::stop_playing() [private]`

We need to keep in sync with `perfdit's stop_playing()`... wish we could call it directly. Well, now we go to the source, calling `perform::stop_playing()`.

7.34.1.8 `void seq64::perfroll::draw_sequence_on(Glib::RefPtr< Gdk::Drawable > a_draw, int a_sequence) [private]`

Statement nesting from hell!

7.34.1.9 `void seq64::perfroll::on_realize() [private]`

Calls the base-class version first.

Then it allocates the additional resources need, that couldn't be initialized in the constructor, and makes some connections.

7.34.1.10 `bool seq64::perfroll::on_button_press_event(GdkEventButton * ev) [private]`

This gives us Seq24 versus Fruity behavior.

7.34.1.11 `bool seq64::perfroll::on_button_release_event(GdkEventButton * ev) [private]`

This gives us Seq24 versus Fruity behavior.

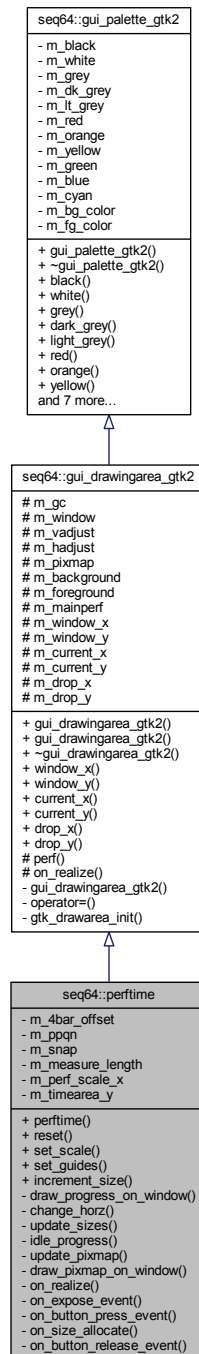
7.34.1.12 `bool seq64::perfroll::on_key_press_event(GdkEventKey * ev) [private]`

If we don't check the event type first, then the `ev->keyval` value is something weird like 65507.

7.35 seq64::perftime Class Reference

This class implements drawing the piano time at the top of the "performance window" (the "song editor").

Inheritance diagram for seq64::perftime:



Public Member Functions

- [perftime](#) ([perform](#) &[perf](#), `Gtk::Adjustment` &`hadjust`)
Principal constructor.
- void [set_guides](#) (int snap, int measure)
Sets the snap value and the measure-length members.
- void [increment_size](#) ()

This function does nothing.

Private Member Functions

- void `change_horz` ()
Change the `m_4bar_offset` and queue a draw operation.
- void `update_sizes` ()
This function does nothing.
- int `idle_progress` ()
This function just returns true.
- void `update_pixmap` ()
This function does nothing.
- void `draw_pixmap_on_window` ()
This function does nothing.
- void `on_realize` ()
Implements the on-realization event, then allocates some resources the could not be allocated in the constructor.
- bool `on_expose_event` (GdkEventExpose *ev)
Implements the on-expose event.
- bool `on_button_press_event` (GdkEventButton *ev)
Implement the button-press event.
- void `on_size_allocate` (Gtk::Allocation &r)
Implements a size-allocation event.
- bool `on_button_release_event` (GdkEventButton *)
This button-release handler does nothing.

Additional Inherited Members

7.35.1 Constructor & Destructor Documentation

7.35.1.1 seq64::perftime::perftime (perform & p, Gtk::Adjustment & hadjust)

In the constructor you can only allocate colors; `get_window()` returns 0 because we have not been realized.

Note

Note that we still have to use a global constant in the base-class constructor; we cannot assign it to the corresponding member beforehand.

7.35.2 Member Function Documentation

7.35.2.1 void seq64::perftime::on_realize () [private]

It is important to call the base-class version of this function.

7.35.2.2 bool seq64::perftime::on_expose_event (GdkEventExpose * ev) [private]

Note

The `perfedit` object is created early on. When brought on-screen from `mainwnd` (the main window), first, `perftime::on_realize()` is called, then this event is called.

It crashes trying to set the foreground color.

7.36 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*

Private Member Functions

- bool `make_directory` (const std::string &pathname) const
- *An internal function to ensure that the ~/.config/sequencer64 directory exists.*

Private Attributes

- `std::string m_filename`

Provides the name of current MIDI file.

7.36.1 Member Function Documentation

7.36.1.1 `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 or could not be created, then an empty string is returned.

7.36.1.2 `bool rc_settings::make_directory (const std::string & pathname) const [private]`

This function is actually a little more general than that, but it is not sufficiently general, in general.

Parameters

<i>pathname</i>	Provides the name of the path to create. The parent directory of the final directory must already exist.
-----------------	--

Returns

Returns true if the path-name exists.

7.37 `seq64::gui_drawingarea_gtk2::rect` Struct Reference

A small helper structure representing a rectangle.

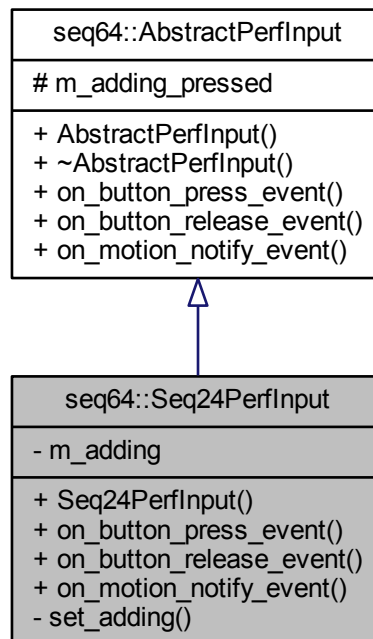
7.38 `seq64::rect` Class Reference

A small helper class representing a rectangle.

7.39 `seq64::Seq24PerfInput` Class Reference

Implements the default performance input characteristics of this application.

Inheritance diagram for seq64::Seq24PerfInput:



Public Member Functions

- bool `on_button_press_event` (GdkEventButton *a_ev, [perfdroll](#) &roll)
Handles the normal variety of button-press event.
- bool `on_button_release_event` (GdkEventButton *a_ev, [perfdroll](#) &roll)
Handles various button-release events.
- bool `on_motion_notify_event` (GdkEventMotion *a_ev, [perfdroll](#) &roll)
Handles the normal motion-notify event.

Private Member Functions

- void `set_adding` (bool a_adding, [perfdroll](#) &roll)
A popup menu (which one?) calls this.

7.39.1 Member Function Documentation

7.39.1.1 bool `seq64::Seq24PerfInput::on_button_press_event` (GdkEventButton * a_ev, [perfdroll](#) & roll) [virtual]

Is there any easy way to use ctrl-left-click as the middle button here?

Implements [seq64::AbstractPerfInput](#).

7.39.1.2 `bool seq64::Seq24PerfInput::on_button_release_event (GdkEventButton * a_ev, perfroll & roll) [virtual]`

Any use for the middle-button or ctrl-left-click we can add?

Implements [seq64::AbstractPerfInput](#).

7.39.1.3 `void seq64::Seq24PerfInput::set_adding (bool adding, perfroll & roll) [private]`

What does it mean?

7.40 seq64::Seq24SeqEventInput Struct Reference

This structure implement the normal interaction methods for Seq24.

Public Member Functions

- [Seq24SeqEventInput](#) ()
Default constructor.
- void [set_adding](#) (bool a_adding, [sequevent](#) &ths)
Changes the mouse cursor to a pencil or a left pointer in the given sequevent aobject, depending on the first parameter.
- bool [on_button_press_event](#) (GdkEventButton *a_ev, [sequevent](#) &ths)
Implements the on-button-press event callback.
- bool [on_button_release_event](#) (GdkEventButton *a_ev, [sequevent](#) &ths)
Implements the on-button-release callback.
- bool [on_motion_notify_event](#) (GdkEventMotion *a_ev, [sequevent](#) &ths)
Implements the on-motion-notify event.

7.40.1 Member Function Documentation

7.40.1.1 `void seq64::Seq24SeqEventInput::set_adding (bool adding, sequevent & segev)`

Modifies m_adding as well.

7.40.1.2 `bool seq64::Seq24SeqEventInput::on_button_press_event (GdkEventButton * a_ev, sequevent & segev)`

Set values for dragging, then reset box that holds dirty redraw spot. Needs update.

segev.m_seq.unselect(); ???????

7.41 seq64::Seq24SeqRollInput Struct Reference

Implements the Seq24 mouse interaction paradigm for the seqroll.

Public Member Functions

- [Seq24SeqRollInput](#) ()
Default constructor.
- void [set_adding](#) (bool a_adding, [seqroll](#) &ths)
Changes the mouse cursor pixmap according to whether a note is being added or not.

- bool [on_button_press_event](#) (GdkEventButton *a_ev, [seqroll](#) &ths)

Implements the on-button-press event handling for the Seq24 style of mouse interaction.

- bool [on_button_release_event](#) (GdkEventButton *a_ev, [seqroll](#) &ths)

Implements the on-button-release event handling for the Seq24 style of mouse interaction.

- bool [on_motion_notify_event](#) (GdkEventMotion *a_ev, [seqroll](#) &ths)

Implements the on-motion-notify event handling for the Seq24 style of mouse interaction.

7.41.1 Member Function Documentation

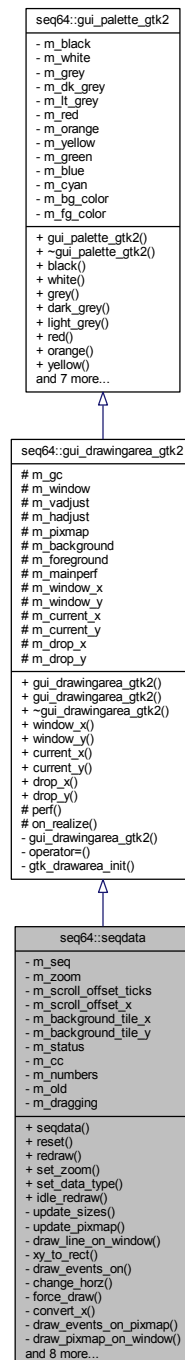
7.41.1.1 void seq64::Seq24SeqRollInput::set_adding (bool *a_adding*, [seqroll](#) & *sroll*)

(Which?) popup menu calls this. It is actually a right click, I think.

7.42 seq64::seqdata Class Reference

This class supports drawing piano-roll events on a window.

Inheritance diagram for seq64::seqdata:



Public Member Functions

- [seqdata](#) ([sequence](#) &seq, [perform](#) &p, int zoom, Gtk::Adjustment &hadjust)

Principal constructor.

- void [reset](#) ()

This function calls [update_size](#)().

- void [redraw](#) ()

- Updates the pixmap and queues up a redraw operation.*

 - void [set_zoom](#) (int a_zoom)

Sets the zoom to the given value and resets the view via the reset function.

 - void [set_data_type](#) (unsigned char a_status, unsigned char a_control)
- Sets the status to the given value, and the control to the optional given value, which defaults to 0, then calls [redraw\(\)](#).*
- int [idle_redraw](#) ()
- Draws events on this object's built-in window and pixmap.*

Private Member Functions

- void [update_sizes](#) ()
- Updates the sizes in the pixmap if the view is realized, and queues up a draw operation.*
- void [update_pixmap](#) ()
- Simply calls [draw_events_on_pixmap\(\)](#).*
- void [draw_line_on_window](#) ()
- Draws on vertical line on...*
- void [xy_to_rect](#) (int a_x1, int a_y1, int a_x2, int a_y2, int &r_x, int &r_y, int &r_w, int &r_h)
- This function takes two points, and returns an Xwin rectangle, returned via the last four parameters.*
- void [draw_events_on](#) (Glib::RefPtr< Gdk::Drawable > a_draw)
- Draws events on the given drawable object.*
- void [change_horz](#) ()
- Change the scrolling offset on the x-axis, and redraw.*
- void [force_draw](#) ()
- Force a redraw.*
- void [convert_x](#) (int x, long &tick)
- This function takes screen coordinates, and gives the horizontal tick value based on the current zoom, returned via the second parameter.*
- void [draw_events_on_pixmap](#) ()
- Simply calls [draw_events_on\(\)](#) for this object's built-in pixmap.*
- void [draw_pixmap_on_window](#) ()
- Simply queues up a draw operation.*
- void [on_realize](#) ()
- Implements the on-realization event, by calling the base-class version and then allocating the resources that could not be allocated in the constructor.*
- bool [on_expose_event](#) (GdkEventExpose *a_ev)
- Implements the on-expose event.*
- bool [on_button_press_event](#) (GdkEventButton *a_ev)
- Implement a button-press event.*
- bool [on_button_release_event](#) (GdkEventButton *a_ev)
- Implement a button-release event.*
- bool [on_motion_notify_event](#) (GdkEventMotion *a_p0)
- Handles a motion-notify event.*
- bool [on_leave_notify_event](#) (GdkEventCrossing *p0)
- Handles an on-leave notification event.*
- bool [on_scroll_event](#) (GdkEventScroll *a_ev)
- Implements the on-scroll event.*
- void [on_size_allocate](#) (Gtk::Allocation &)
- Handle a size-allocation event.*

Private Attributes

- int [m_zoom](#)
one pixel == m_zoom ticks
- unsigned char [m_status](#)
What is the data window currently editing?

Additional Inherited Members

7.42.1 Constructor & Destructor Documentation

7.42.1.1 `seq64::seqdata::seqdata (sequence & seq, perform & p, int zoom, Gtk::Adjustment & hadjust)`

In the constructor you can only allocate colors, `get_window()` returns 0 because we have not been realized.

7.42.2 Member Function Documentation

7.42.2.1 `void seq64::seqdata::reset ()`

Then, regardless of whether the view is realized, updates the pixmap and queues up a draw operation.

Note

If it weren't for the `is_realized()` condition, we could just call [update_sizes\(\)](#), which does all this anyway.

7.42.2.2 `void seq64::seqdata::redraw () [inline]`

We need to make this an inline function and use it as common code.

7.42.2.3 `void seq64::seqdata::set_zoom (int zoom)`

This begs the question, do we have GUI access to the zoom setting?

7.42.2.4 `int seq64::seqdata::idle_redraw ()`

This drawing is done only if there is no dragging in progress, to guarantee no flicker.

7.42.2.5 `void seq64::seqdata::update_sizes () [private]`

It creates a pixmap with window dimensions given by `m_window_x` and `m_window_y`.

7.42.2.6 `void seq64::seqdata::xy_to_rect (int a_x1, int a_y1, int a_x2, int a_y2, int & r_x, int & r_y, int & r_w, int & r_h) [private]`

It checks the mins/maxes, then fills in x, y, and width, height.

7.42.2.7 `void seq64::seqdata::on_realize () [private]`

It also connects up the [change_horz\(\)](#) function.

7.42.2.8 `bool seq64::seqdata::on_motion_notify_event (GdkEventMotion * a_p0) [private]`

It converts the x,y of the mouse to ticks, then sets the events in the event-data-range, updates the pixmap, draws events in the window, and draws a line on the window.

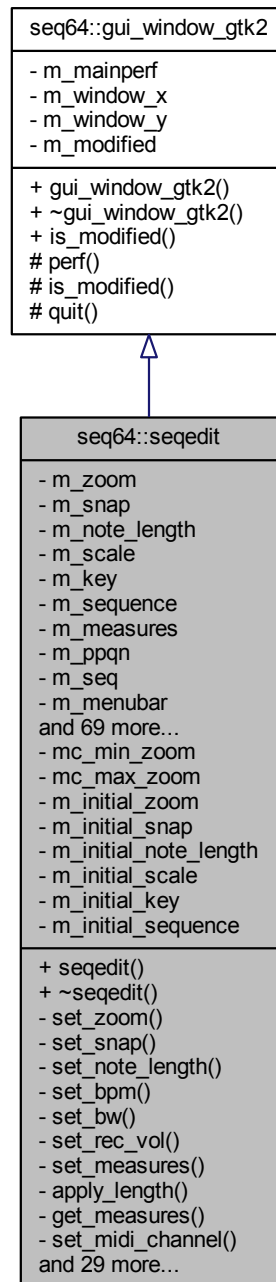
7.42.2.9 `bool seq64::seqdata::on_scroll_event (GdkEventScroll * a_ev) [private]`

This scroll event only handles basic scrolling, without any modifier keys such as GDK_CONTROL_MASK or GDK_SHIFT_MASK.

7.43 seq64::seqedit Class Reference

Implements the Pattern Editor, which has references to:

Inheritance diagram for seq64::seqedit:



Public Member Functions

- `seqedit (sequence &a_seq, perform &a_perf, int pos, int ppqn=SEQ64_USE_DEFAULT_PPQN)`
Connects to a menu item, tells the performance to launch the timer thread.
- `~seqedit ()`
A rote destructor.

Private Member Functions

- void [set_zoom](#) (int a_zoom)
Selects the given zoom value.
- void [set_snap](#) (int a_snap)
Selects the given snap value.
- void [set_note_length](#) (int a_note_length)
Selects the given note-length value.
- void [set_bpm](#) (int a_beats_per_measure)
Set the bpm (beats per measure) value, using the given parameter, and some internal values passed to [apply_length\(\)](#).
- void [set_bw](#) (int a_beat_width)
Set the bw (beat width) value, using the given parameter, and some internal values passed to [apply_length\(\)](#).
- void [set_rec_vol](#) (int a_rec_vol)
Passes the given parameter to [sequence::set_rec_vol\(\)](#).
- void [set_measures](#) (int a_length_measures)
Set the measures value, using the given parameter, and some internal values passed to [apply_length\(\)](#).
- void [apply_length](#) (int a_bpm, int a_bw, int a_measures)
Sets the length based on the three given parameters.
- long [get_measures](#) ()
Calculates the measures value based on the bpm (beats per measure), ppqn (parts per quarter note), and bw (beat width) values, and returns the resultant measures value.
- void [set_midi_channel](#) (int a_midichannel)
Selects the given MIDI channel parameter in the main sequence object, so that it will use that channel.
- void [set_midi_bus](#) (int a_midibus)
Selects the given MIDI buss parameter in the main sequence object, so that it will use that buss.
- void [set_scale](#) (int a_scale)
Selects the given scale value.
- void [set_key](#) (int a_note)
Selects the given key (signature) value.
- void [set_background_sequence](#) (int a_seq)
Draws the given background sequence on the Pattern editor so that the musician has something to see that can be played against.
- void [name_change_callback](#) ()
Set the name for the main sequence to this object's entry name.
- void [play_change_callback](#) ()
Passes the play status to the sequence object.
- void [record_change_callback](#) ()
Passes the recording status to the sequence object.
- void [q_rec_change_callback](#) ()
Passes the quantized-recording status to the sequence object.
- void [thru_change_callback](#) ()
Passes the MIDI Thru status to the sequence object.
- void [undo_callback](#) ()
Pops an undo operation from the sequence object, and then tell the segroll, seqtime, seqdata, and sequevent objects to redraw.
- void [redo_callback](#) ()
Pops a redo operation from the sequence object, and then tell the segroll, seqtime, seqdata, and sequevent objects to redraw.
- void [set_data_type](#) (unsigned char a_status, unsigned char a_control=0)
Sets the data type based on the given parameters.
- void [fill_top_bar](#) ()

This function inserts the user-interface items into the top bar or panel of the pattern editor; this bar has two rows of user interface elements.

- void `create_menus` ()
Creates the various menus by pushing menu elements into the menus.
- void `popup_menu` (Gtk::Menu *a_menu)
Pops up the given pop-up menu.
- void `popup_event_menu` ()
Populates the event-selection menu that drops from the "Event" button in the bottom row of the Pattern editor.
- void `popup_midibus_menu` ()
Populates the MIDI Output buss pop-up menu.
- void `popup_sequence_menu` ()
Populates the "set background sequence" menu (drops from the button that has some note-bars on it at the right of the second row of the top bar).
- void `popup_tool_menu` ()
Sets up the pop-up menus that are brought up by pressing the Tools button, which shows a hammer image.
- void `popup_midich_menu` ()
Populates the MIDI Channel pop-up menu.
- Gtk::Image * `create_menu_image` (bool a_state=false)
Sets the manu pixmap depending on the given state, where true is a full menu (black background), and empty menu (gray background).
- bool `timeout` ()
Update the window after a time out, based on dirtiness and on playback progress.
- void `do_action` (int a_action, int a_var)
Implements the actions brought forth from the Tools (hammer) button.
- void `on_realize` ()
On realization, calls the base-class version, and connects the redraw timeout signal, timed at c_redraw_ms.
- bool `on_delete_event` (GdkEventAny *a_event)
Handles an on-delete event.
- bool `on_scroll_event` (GdkEventScroll *a_ev)
Handles an on-scroll event.
- bool `on_key_press_event` (GdkEventKey *a_ev)
Handles a key-press event.

Private Attributes

- int `m_zoom`
Provides the zoom values: 0 1 2 3 4, and 1, 2, 4, 8, 16.
- int `m_snap`
Use in setting the snap-to in pulses, off = 1.
- int `m_scale`
Settings for the music scale and key.
- Gtk::Menu * `m_menu_length`
Provides the length in measures.
- Gtk::Menu * `m_menu_bpm`
These member provide the time signature, beats per measure, and beat width menus.
- unsigned char `m_editing_status`
Indicates what is the data window currently editing?

Static Private Attributes

- static const int `mc_min_zoom`
Static data members.

Additional Inherited Members

7.43.1 Detailed Description

- perform
- seqroll
- seqkeys
- seqdata
- seqtime
- seqevent
- sequence

This class has a metric ton of user-interface objects and other members.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 `seq64::seqedit::seqedit (sequence & seq, perform & p, int pos, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

But this is an unused, empty function.

`void seqedit::menu_action_quantise () { }` Principal constructor.

Todo Offload most of the work into an initialization function like options does; make the sequence and perform parameters references.

7.43.3 Member Function Documentation

7.43.3.1 `void seq64::seqedit::set_zoom (int a_zoom) [private]`

It is passed to the seqroll, seqtime, seqdata, and seqevent objects, as well.

7.43.3.2 `void seq64::seqedit::set_snap (int a_snap) [private]`

It is passed to the seqroll, seqevent, and sequence objects, as well.

7.43.3.3 `void seq64::seqedit::set_note_length (int a_note_length) [private]`

It is passed to the seqroll object, as well.

7.43.3.4 `void seq64::seqedit::apply_length (int a_bpm, int a_bw, int a_measures) [private]`

Then the seqroll, seqtime, seqdata, and seqevent objects are reset().

7.43.3.5 `long seq64::seqedit::get_measures () [private]`

Todo Create a `sequence::set_units()` function or a `sequence::get_measures()` function to forward to.

7.43.3.6 `void seq64::seqedit::set_scale (int a_scale) [private]`

It is passed to the seqroll and seqkeys objects, as well.

7.43.3.7 `void seq64::seqedit::set_key (int a_note) [private]`

It is passed to the seqroll and seqkeys objects, as well.

7.43.3.8 `void seq64::seqedit::set_background_sequence (int a_seq) [private]`

Todo Make the sequence pointer a reference.

7.43.3.9 `void seq64::seqedit::name_change_callback () [private]`

That name is the name the user has given to the sequence being edited.

7.43.3.10 `void seq64::seqedit::set_data_type (unsigned char a_status, unsigned char a_control = 0) [private]`

To be determined.

7.43.3.11 `void seq64::seqedit::popup_event_menu () [private]`

This menu has a large number of items. I think they are filled in in code, but can also be loaded from `~/seq24usr`. To be determined. Create the 8 sub-menus for the various ranges of controller changes, shown 16 per sub-menu.

7.43.3.12 `void seq64::seqedit::popup_midibus_menu () [private]`

The MIDI busses are obtained by getting the mastermidibus object, and iterating through the busses that it contains.

7.43.3.13 `void seq64::seqedit::popup_sequence_menu () [private]`

It is populated with an "Off" menu entry, and a second "[0]" menu entry that pulls up a drop-down menu of all of the patterns/sequences that are present in the MIDI file.

7.43.3.14 `void seq64::seqedit::popup_tool_menu () [private]`

This button shows three sub-menus that need to be filled in by this function. All the functions accessed here seem to be implemented by the `do_action()` function.

7.43.3.15 `void seq64::seqedit::do_action (int a_action, int a_var) [private]`

Note that the `push_undo()` calls push all of the current events (in `sequence::m_events`) onto the stack (as a single entry).

7.43.3.16 `bool seq64::seqedit::on_delete_event (GdkEventAny * a_event) [private]`

It tells the sequence to stop recording, tells the perform object's mastermidibus to stop processing input, and sets the sequence object's editing flag to false.

Warning

This function also calls "delete this"!

Returns

Always returns false.

7.43.4 Field Documentation

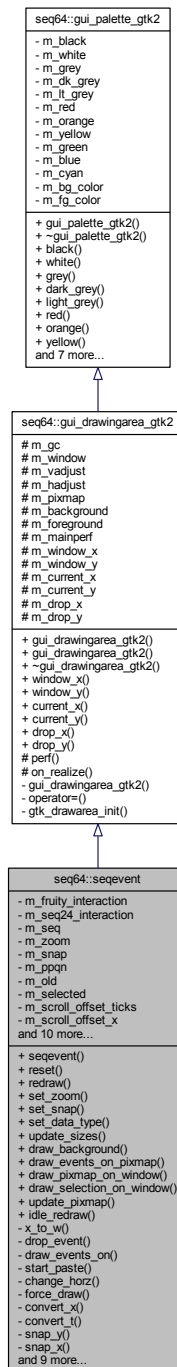
7.43.4.1 `const int seq64::segedit::mc_min_zoom` `[static], [private]`

These items apply to all of the instances of segedit.

7.44 seq64::sequevent Class Reference

Implements the piano event drawing area.

Inheritance diagram for seq64::sequevent:



Public Member Functions

- `sequevent` (`sequence` &seq, `perform` &p, int zoom, int snap, `seqdata` &seqdata_wid, `Gtk::Adjustment` &hadjust, int ppqn=SEQ64_USE_DEFAULT_PPQN)

Principal constructor.

- void `reset` ()

This function basically resets the whole widget as if it was realized again.

- void [redraw](#) ()
Adjusts the scrolling offset for ticks, updates the pixmap, and draws it on the window.
- void [set_zoom](#) (int a_zoom)
Sets zoom to the given value, and resets if the value ended up being changed.
- void [set_snap](#) (int a_snap)
'Setter' function for member m_snap
- void [set_data_type](#) (unsigned char a_status, unsigned char a_control)
Sets the status to the given parameter, and the CC value to the given optional control parameter, which defaults to 0.
- void [update_sizes](#) ()
If the window is realized, this function creates a pixmap with window dimensions, then updates the pixmap, and queues up a redraw.
- void [draw_background](#) ()
This function updates the background.
- void [draw_events_on_pixmap](#) ()
This function fills the main pixmap with events.
- void [draw_pixmap_on_window](#) ()
This function currently just queues up a draw operation for the pixmap.
- void [draw_selection_on_window](#) ()
Draw the selected events on the window.
- void [update_pixmap](#) ()
Redraws the background pixmap on the main pixmap, then puts the events on.
- int [idle_redraw](#) ()
Implements redraw while idling.

Private Member Functions

- void [x_to_w](#) (int a_x1, int a_x2, int &a_x, int &a_w)
This function checks the mins / maxes.
- void [drop_event](#) (long a_tick)
Drops (adds) an event at the given tick.
- void [draw_events_on](#) (Glib::RefPtr< Gdk::Drawable > a_draw)
Draws events on the given drawable object.
- void [start_paste](#) ()
Starts a paste operation.
- void [change_horz](#) ()
Changes the horizontal scrolling offset for ticks, then updates the pixmap and forces a redraw.
- void [force_draw](#) ()
Forces a draw on the current drawable area of the window.
- void [convert_x](#) (int x, long &tick)
Takes the screen x coordinate, multiplies it by the current zoom, and returns the tick value in the given parameter.
- void [convert_t](#) (long ticks, int &x)
Converts the given tick value to an x coordinate, based on the zoom, and returns it via the second parameter.
- void [snap_y](#) (int &y)
This function performs a 'snap' on y.
- void [snap_x](#) (int &a_x)
This function performs a 'snap' on x.
- void [on_realize](#) ()
Implements the on-realize callback.
- bool [on_expose_event](#) (GdkEventExpose *a_ev)
Implements the on-expose event callback.
- bool [on_button_press_event](#) (GdkEventButton *a_ev)

- Implements the on-button-press event callback.*
- bool [on_button_release_event](#) (GdkEventButton *a_ev)
Implements the on-button-release event callback.
- bool [on_motion_notify_event](#) (GdkEventMotion *a_ev)
Implements the on-motion-notify event callback.
- bool [on_focus_in_event](#) (GdkEventFocus *)
Responds to a focus event by setting the HAS_FOCUS flag.
- bool [on_focus_out_event](#) (GdkEventFocus *)
Responds to a unfocus event by resetting the HAS_FOCUS flag.
- bool [on_key_press_event](#) (GdkEventKey *a_p0)
Implements the key-press event callback function.
- void [on_size_allocate](#) (Gtk::Allocation &)
Implements the on-size-allocate event callback.

Private Attributes

- FruitySeqEventInput [m_fruity_interaction](#)
Why should we need both at the same time? Just load the one that is specified in the configuration.
- int [m_zoom](#)
Zoom setting, means that one pixel == m_zoom ticks.
- bool [m_selecting](#)
Used when highlighting a bunch of events.
- unsigned char [m_status](#)
Indicates what is the data window currently editing?

Additional Inherited Members

7.44.1 Member Function Documentation

7.44.1.1 void seq64::seqevent::set_snap (int a_snap) [inline]

Simply sets the snap member.

7.44.1.2 void seq64::seqevent::set_data_type (unsigned char status, unsigned char control = 0)

Then redraws.

7.44.1.3 void seq64::seqevent::update_sizes ()

This ends up filling the background with dotted lines, etc.

7.44.1.4 void seq64::seqevent::draw_background ()

It sets the foreground to white, draws the rectangle.

7.44.1.5 void seq64::seqevent::draw_pixmap_on_window ()

Old comments:

```
It then tells event to do the same.
We changed something on this window, and chances are we need to
update the event widget as well and update our velocity window.
```

7.44.1.6 `int seq64::sequevent::idle_redraw ()`

Who calls this routine?

7.44.1.7 `void seq64::sequevent::x_to_w (int a_x1, int a_x2, int & a_x, int & a_w) [private]`

Then it fills in x and the width.

7.44.1.8 `void seq64::sequevent::drop_event (long a_tick) [private]`

It sets the first byte properly for after-touch, program-change, channel-pressure, and pitch-wheel. The type of event is determined by m_status.

7.44.1.9 `void seq64::sequevent::start_paste () [private]`

It gets the clipboard box that selected elements are in, makes a coordinate conversion, and then, sets the m_↔ selected rectangle to hold the (x,y,w,h) of the selected events.

7.44.1.10 `void seq64::sequevent::convert_x (int x, long & tick) [inline], [private]`

Why not just return it normally?

7.44.1.11 `void seq64::sequevent::convert_t (long ticks, int & x) [inline], [private]`

Why not just return it normally?

7.44.1.12 `void seq64::sequevent::snap_x (int & x) [private]`

- snap = number pulses to snap to
- m_zoom = number of pulses per pixel,

Therefore snap / m_zoom = number pixels to snap to.

7.44.1.13 `void seq64::sequevent::on_realize () [private]`

It calls the base-class version, and then allocates additional resource not allocated in the constructor. Finally, it connects up the change_horz function.

7.44.1.14 `bool seq64::sequevent::on_button_press_event (GdkEventButton * a_ev) [private]`

It distinguishes between the Seq24 and Fruity varieties of mouse interaction.

Odd. In the legacy code, each case fell through to the next case to the "default" case! We will assume for now that this is incorrect.

Note that returning "true" from a Gtkmm event-handler stops the propagation of the event to higher-level widgets. The Fruity and Seq24 event handlers return true, always. In the legacy code, though, the fall-through code caused false to be returned, always. Not sure what effect this had.

7.44.1.15 `bool seq64::segevent::on_button_release_event (GdkEventButton * a_ev) [private]`

It distinguishes between the Seq24 and Fruity varieties of mouse interaction.

Odd. The fruity case fell through to the Seq24 case. We will assume for now that this is correct.

7.44.1.16 `bool seq64::segevent::on_motion_notify_event (GdkEventMotion * a_ev) [private]`

It distinguishes between the Seq24 and Fruity varieties of mouse interaction.

Odd. The fruity case fell through to the Seq24 case. We will assume for now that this is correct.

7.44.1.17 `bool seq64::segevent::on_key_press_event (GdkEventKey * a_p0) [private]`

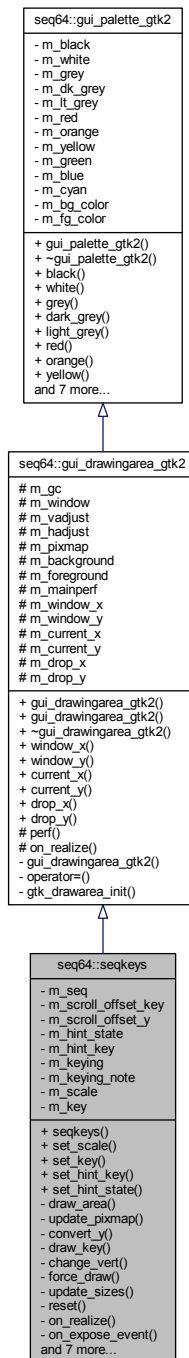
It handles deleted a selection via the Backspace or Delete keys, cut via Ctrl-X, copy via Ctrl-C, paste via Ctrl-V, and undo via Ctrl-Z.

Would be nice to provide redo functionality via Ctrl-Y. :-)

7.45 `seq64::seqkeys` Class Reference

This class implements the left side piano of the pattern/sequence editor.

Inheritance diagram for seq64::seqkeys:



Public Member Functions

- [seqkeys](#) ([sequence](#) &seq, [perform](#) &p, Gtk::Adjustment &vadjust)
Principal constructor.
- void [set_scale](#) (int a_scale)
Sets the musical scale, then resets.
- void [set_key](#) (int a_key)

- *Sets the musical key, then resets.*
- void [set_hint_key](#) (int a_key)
Sets a key to grey so that it can serve as a scale hint.
- void [set_hint_state](#) (bool a_state)
Sets the hint state to the given value.

Private Member Functions

- void [draw_area](#) ()
Draws the updated pixmap on the drawable area of the window where the keys' location is hardwired.
- void [update_pixmap](#) ()
Updates the pixmaps to prepare it for the next draw operation.
- void [convert_y](#) (int a_y, int &a_note)
Takes the screen y coordinate, and returns the note value in the second parameter.
- void [draw_key](#) (int a_key, bool a_state)
Draws the given key according to the given state.
- void [change_vert](#) ()
Changes the y offset of the scrolling, and the forces a draw.
- void [force_draw](#) ()
Forces a draw operation on the whole window.
- void [reset](#) ()
Resetting the keys view updates the pixmap and queues up a draw operation.
- void [on_realize](#) ()
Implements the on-realize event.
- bool [on_expose_event](#) (GdkEventExpose *a_ev)
Implements the on-expose event, by drawing on the window.
- bool [on_button_press_event](#) (GdkEventButton *a_ev)
Implements the on-button-press event callback.
- bool [on_button_release_event](#) (GdkEventButton *a_ev)
Implements the on-button-release event callback.
- bool [on_motion_notify_event](#) (GdkEventMotion *a_p0)
Implements the on-motion-notify event handler.
- bool [on_enter_notify_event](#) (GdkEventCrossing *p0)
Implements the on-enter notification event handler.
- bool [on_leave_notify_event](#) (GdkEventCrossing *p0)
Implements the on-leave notification event handler.
- bool [on_scroll_event](#) (GdkEventScroll *a_ev)
Implements the on-scroll-event notification event handler.
- void [on_size_allocate](#) (Gtk::Allocation &)
Implements the on-size-allocation notification event handler.

Private Attributes

- bool [m_keying](#)
What is this?

Additional Inherited Members

7.45.1 Member Function Documentation

7.45.1.1 void seq64::seqkeys::set_hint_state (bool state)

Parameters

<i>state</i>	Provides the value for hinting, where true == on, false == off.
--------------	---

7.45.1.2 `void seq64::seqkeys::draw_key (int a_key, bool a_state)` [private]

It accounts for the black keys and the white keys.

Parameters

<i>a_key</i>	The key to be drawn.
<i>a_state</i>	How the key is to be drawn, where false == normal, true == grayed.

7.45.1.3 `void seq64::seqkeys::on_realize ()` [private]

Call the base-class version and then allocates resources that could not be allocated in the constructor. It connects the [change_vert\(\)](#) function and then calls it.

7.45.1.4 `bool seq64::seqkeys::on_button_press_event (GdkEventButton * ev)` [private]

It currently handles only the left button. This button, pressed on the piano keyboard, causes m_keying to be set to true, and the given note to play.

7.45.1.5 `bool seq64::seqkeys::on_button_release_event (GdkEventButton * ev)` [private]

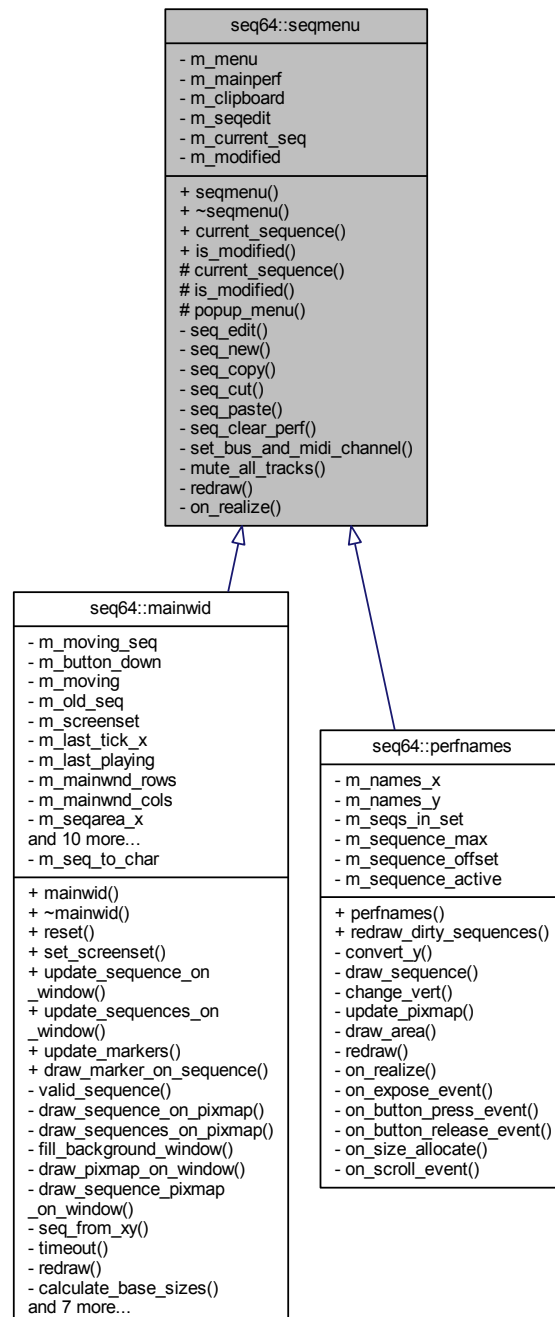
It currently handles only the left button, and only if m_keying is true.

This function is used after pressing on one of the keys on the left-side piano keyboard, to make it play, and turns off the playing of the note.

7.46 seq64::seqmenu Class Reference

This class handles the right-click menu of the sequence slots in the pattern window.

Inheritance diagram for seq64::seqmenu:



Public Member Functions

- [seqmenu](#) ([perform](#) &a_p)
Principal constructor.
- virtual [~seqmenu](#) ()
Provides a rote base-class destructor.
- int [current_sequence](#) () const

- *'Getter' function for member m_current_seq*
- bool [is_modified](#) () const
- *'Getter' function for member m_modified*

Protected Member Functions

- void [current_sequence](#) (int seq)
- *'Setter' function for member m_current_seq*
- void [is_modified](#) (bool flag)
- *'Setter' function for member m_modified*
- void [popup_menu](#) ()
- *This function sets up the File menu entries.*

Private Member Functions

- void [seq_edit](#) ()
- *This menu callback launches the sequence-editor (pattern editor) window.*
- void [seq_new](#) ()
- *This function sets the new sequence into the perform object, a bit prematurely, though.*
- void [seq_copy](#) ()
- *Copies the selected (current) sequence to the clipboard sequence.*
- void [seq_cut](#) ()
- *Deletes the selected (current) sequence and copies it to the clipboard sequence, if it is not in edit mode.*
- void [seq_paste](#) ()
- *Pastes the sequence clipboard into the current sequence, if the current sequence slot is not active.*
- void [seq_clear_perf](#) ()
- *If the current sequence is active, this function pushes a trigger undo in the main perform object, clears its sequence triggers for the current sequence, and sets the dirty flag of the sequence.*
- void [set_bus_and_midi_channel](#) (int a_bus, int a_ch)
- *Sets up the bus, MIDI channel, and dirtiness flag of the current sequence in the main perform object, as per the give parameters.*
- void [mute_all_tracks](#) ()
- *Mutes all tracks in the main perform object.*

Private Attributes

- [seqedit](#) * [m_seqedit](#)
- *Change Note Added by Chris on 2015-08-02 based on compiler warnings and a comment warning in the [seq_edit\(\)](#) function.*

7.46.1 Detailed Description

It is an abstract base class.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 seq64::seqmenu::seqmenu (perform & a_p)

Apart from filling in some of the members, this function initializes the clipboard, so that we don't get a crash on a paste with no previous copy.

7.46.2.2 seq64::seqmenu::~~seqmenu () [virtual]

A rote destructor.

This is necessary in an abstraction base class.

If we determine that we need to delete the m_seqedit pointer, we can do it here. But that is not likely, because we can have many new seqedit objects in play, because we can edit many at once.

7.46.3 Member Function Documentation

7.46.3.1 void seq64::seqmenu::seq_edit () [private]

If it is already open for that sequence, this function just raises it.

Note that the m_seqedit member to which we save the new pointer is currently there just to avoid a compiler warning.

Also, if a new sequences is created, we set the m_modified flag to true, even though the sequence might later be deleted. Too much modification to keep track of!

7.46.3.2 void seq64::seqmenu::seq_copy () [private]

Todo Can be offloaded to a perform member function that accepts a sequence clipboard non-const reference parameter.

7.46.3.3 void seq64::seqmenu::seq_cut () [private]

Todo A lot of seq_cut() can be offloaded to a (new) perform member function that takes a sequence clipboard non-const reference parameter.

7.46.3.4 void seq64::seqmenu::seq_paste () [private]

Then it sets the dirty flag for the destination sequence.

Todo All of seq_paste() can be offloaded to a (new) perform member function with a const clipboard reference parameter.

7.46.3.5 void seq64::seqmenu::seq_clear_perf () [private]

Todo All of seq_paste() can be offloaded to a (new) perform member function.

7.46.4 Field Documentation

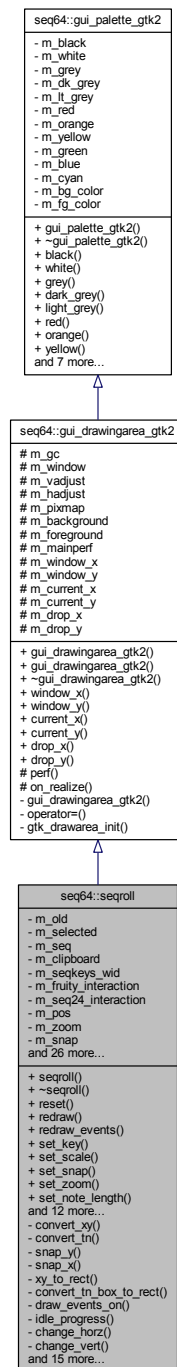
7.46.4.1 seqedit* seq64::seqmenu::m_seqedit [private]

We'll save the result of that function here, and will let valgrind tell us later if Gtkmm takes care of it.

7.47 seq64::seqroll Class Reference

Implements the piano roll section of the pattern editor.

Inheritance diagram for seq64::seqroll:



Public Member Functions

- `seqroll` (`perform` &`perf`, `sequence` &`seq`, `int` zoom, `int` snap, `seqkeys` &`seqkeys_wid`, `int` pos, `Gtk::Adjustment` &`hadjust`, `Gtk::Adjustment` &`vadjust`, `int` ppqn=`SEQ64_USE_DEFAULT_PPQN`)

Principal constructor.

- `~seqroll` ()

Provides a destructor to delete allocated objects.

- void `reset` ()
This function basically resets the whole widget as if it was realized again.
- void `redraw` ()
Redraws unless `m_ignore_redraw` is true.
- void `redraw_events` ()
Redraws events unless `m_ignore_redraw` is true.
- void `set_key` (int key)
Sets the music key to the given value, and then resets the view.
- void `set_scale` (int scale)
Sets the music scale to the given value, and then resets the view.
- void `set_snap` (int snap)
Sets the snap to the given value, and then resets the view.
- void `set_zoom` (int zoom)
Sets the zoom to the given value, and then resets the view.
- void `set_note_length` (int note_length)
'Setter' function for member `m_note_length`
- void `set_ignore_redraw` (bool ignore)
'Setter' function for member `m_ignore_redraw`
- void `set_data_type` (unsigned char status, unsigned char control)
Sets the status to the given parameter, and the CC value to the given optional control parameter, which defaults to 0.
- void `set_background_sequence` (bool state, int seq)
This function sets the given sequence onto the piano roll of the pattern editor, so that the musician can have another pattern to play against.
- void `update_pixmap` ()
This function draws the background pixmap on the main pixmap, and then draws the events on it.
- void `update_sizes` ()
Update the sizes of items based on zoom, PPQN, BPM, BW (beat width) and more.
- void `update_background` ()
Updates the background of this window.
- void `draw_background_on_pixmap` ()
Draws the main pixmap.
- void `draw_events_on_pixmap` ()
Fills the main pixmap with events.
- void `draw_selection_on_window` ()
Draws the current selection on the main window.
- void `draw_progress_on_window` ()
Draw a progress line on the window.
- int `idle_redraw` ()
Draw the events on the main window and on the pixmap.
- void `start_paste` ()
Starts a paste operation.

Private Member Functions

- void `convert_tn` (long ticks, int note, int &x, int &y)
This function takes the given note and tick, and returns the screen coordinates via the pointer parameters.
- void `snap_x` (int &x)
Performs a 'snap' operation on the x coordinate.
- void `xy_to_rect` (int x1, int y1, int x2, int y2, int &x, int &y, int &w, int &h)
This function checks the mins / maxes, and then fills in the x, y, width, and height values.
- void `convert_tn_box_to_rect` (long tick_s, long tick_f, int note_h, int note_l, int &x, int &y, int &w, int &h)

- Converts a tick/note box to an x/y rectangle.*
- void [draw_events_on](#) (Glib::RefPtr< Gdk::Drawable > draw)
 - Draws events on the given drawable area.*
- void [change_horz](#) ()
 - Change the horizontal scrolling offset and redraw.*
- void [change_vert](#) ()
 - Change the vertical scrolling offset and redraw.*
- void [force_draw](#) ()
 - Set the pixmap into the window and then draws the selection on it.*
- void [on_realize](#) ()
 - Implements the on-realize event handling.*
- bool [on_expose_event](#) (GdkEventExpose *ev)
 - Implements the on-expose event handling.*
- bool [on_button_press_event](#) (GdkEventButton *ev)
 - Implements the on-button-press event handling.*
- bool [on_button_release_event](#) (GdkEventButton *ev)
 - Implements the on-button-release event handling.*
- bool [on_motion_notify_event](#) (GdkEventMotion *ev)
 - Implements the on-motion-notify event handling.*
- bool [on_focus_in_event](#) (GdkEventFocus *)
 - Implements the on-focus event handling.*
- bool [on_focus_out_event](#) (GdkEventFocus *)
 - Implements the on-unfocus event handling.*
- bool [on_key_press_event](#) (GdkEventKey *ev)
 - Implements the on-key-press event handling.*
- bool [on_scroll_event](#) (GdkEventScroll *a_ev)
 - Implements the on-scroll event handling.*
- void [on_size_allocate](#) (Gtk::Allocation &)
 - Implements the on-size-allocate event handling.*
- bool [on_leave_notify_event](#) (GdkEventCrossing *p0)
 - Implements the on-leave-notify event handling.*
- bool [on_enter_notify_event](#) (GdkEventCrossing *p0)
 - Implements the on-enter-notify event handling.*

Private Attributes

- int [m_zoom](#)
 - one pixel == m_zoom ticks**
- unsigned char [m_status](#)
 - Indicates what is the data window currently editing.*
- bool [m_selecting](#)
 - When highlighting a bunch of events.*
- int [m_move_delta_x](#)
 - Tells where the dragging started.*

Additional Inherited Members

7.47.1 Member Function Documentation

7.47.1.1 void seq64::seqroll::reset ()

It's almost identical to the [change_horz\(\)](#) function!

7.47.1.2 `void seq64::seqroll::set_data_type (unsigned char status, unsigned char control)`

Unlike the same function in `sequevent`, this version does not redraw.

7.47.1.3 `void seq64::seqroll::set_background_sequence (bool state, int seq)`

The `a_state` parameter sets the boolean `m_drawing_background_seq`.

7.47.1.4 `void seq64::seqroll::draw_events_on_pixmap ()`

Just calls [draw_events_on\(\)](#).

7.47.1.5 `void seq64::seqroll::convert_tn (long a_ticks, int a_note, int & a_x, int & a_y)` [private]

This function is the "inverse" of `convert_xy()`.

7.47.1.6 `void seq64::seqroll::snap_x (int & x)` [private]

This function is similar to `snap_y()`, but it calculates a modulo value from the snap and zoom settings.

- `m_snap` = number pulses to snap to
- `m_zoom` = number of pulses per pixel

Therefore, `m_snap / m_zoom` = number pixels to snap to.

7.47.1.7 `bool seq64::seqroll::on_key_press_event (GdkEventKey * a_p0)` [private]

The start/end key may be the same key (i.e. SPACEBAR). Allow toggling when the same key is mapped to both triggers (i.e. SPACEBAR).

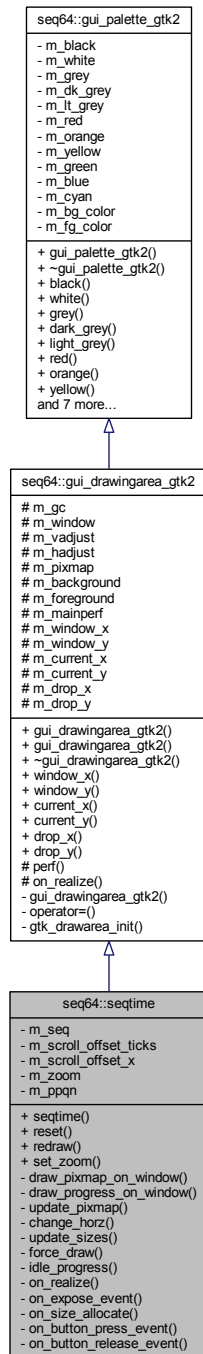
7.47.1.8 `bool seq64::seqroll::on_scroll_event (GdkEventScroll * a_ev)` [private]

This scroll event only handles basic scrolling without any modifier keys such as `GDK_CONTROL_MASK` or `GDK_SHIFT_MASK`.

7.48 seq64::seqtime Class Reference

This class implements the piano time, whatever that is.

Inheritance diagram for seq64::seqtime:



Public Member Functions

- void [set_zoom](#) (int zoom)

Sets the zoom to the given value and resets the window.

Private Member Functions

- bool [idle_progress](#) ()
Simply returns true.
- bool [on_button_press_event](#) (GdkEventButton *)
Implements the on-button-press event handler.
- bool [on_button_release_event](#) (GdkEventButton *)
Implements the on-button-release event handler.

Private Attributes

- int [m_zoom](#)
one pixel == m_zoom ticks

Additional Inherited Members

7.48.1 Member Function Documentation

7.48.1.1 bool [seq64::seqtime::on_button_press_event](#) (GdkEventButton *) [inline],[private]

Simply returns false.

7.48.1.2 bool [seq64::seqtime::on_button_release_event](#) (GdkEventButton *) [inline],[private]

Simply returns false.

7.49 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_select_one](#),
 [e_is_selected](#),
 [e_would_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](#) (int ppqn=SEQ64_USE_DEFAULT_PPQN)
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 copied 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 ¬e)
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 [selected_trigger_start](#) ()
Gets the selected trigger's start tick.
- long [selected_trigger_end](#) ()
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 ¬e_h, long &tick_f, int ¬e_l)

Returns the 'box' of the selected items.
- void `get_clipboard_box` (long &tick_s, int ¬e_h, long &tick_f, int ¬e_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.

Private Member Functions

- void [put_event_on_bus](#) (event *ev)

Takes an event that this sequence is holding, and places it on the midibus.
- void [set_trigger_offset](#) (long trigger_offset)

Sets m_trigger_offset and wraps it to m_length.
- void [split_trigger](#) (trigger &trig, long split_tick)

Splits the trigger given by the parameter into two triggers.
- void [adjust_trigger_offsets_to_length](#) (long new_len)

Not sure what these diagrams are for yet.
- long [adjust_offset](#) (long offset)

Adjusts the given offset by mod'ing it with m_length and adding m_length if needed, and returning the result.
- void [remove](#) (event_list::iterator i)

A helper function, which does not lock/unlock, so it is unsafe to call without supplying an iterator from the list-event.
- void [remove](#) (event *e)

A helper function, which does not lock/unlock, so it is unsafe to call without supplying an iterator from the list-event.

Private Attributes

- event_list [m_events](#)

This list holds the current pattern/sequence events.
- mutex [m_mutex](#)

Provides locking for the sequence.

Static Private Attributes

- static [event_list m_events_clipboard](#)
A static clipboard for holding pattern/sequence events.

7.49.1 Detailed Description

More members than you can shake a stick at.

7.49.2 Member Enumeration Documentation

7.49.2.1 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.

To select ...

e_select_one To select ...

e_is_selected The events are selected ...

e_would_select The events would be selected ...

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.

7.49.3 Member Function Documentation

7.49.3.1 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. Currently, it does not assign them all, so we should create a `partial_copy()` function to do this work, and use it where it is needed.

Threadsafe

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

Threadsafe

7.49.3.3 void seq64::sequence::push_undo ()

Threadsafe

7.49.3.4 void seq64::sequence::pop_undo ()

Threadsafe

7.49.3.5 void seq64::sequence::pop_redo ()

Threadsafe

7.49.3.6 void seq64::sequence::push_trigger_undo ()

Threadsafe

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

Threadsafe

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

Threadsafe

7.49.3.9 long seq64::sequence::get_bw () const [inline]

Threadsafe

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

Threadsafe

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

Threadsafe

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

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

Parameters

<i>a_p</i>	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.
------------	--

7.49.3.13 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

7.49.3.14 void seq64::sequence::off_queued ()

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

Threadsafe

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

Threadsafe

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

Threadsafe

7.49.3.17 void seq64::sequence::set_quantized_rec (bool a_qr)

Threadsafe

7.49.3.18 void seq64::sequence::set_thru (bool a_r)

Threadsafe

7.49.3.19 bool seq64::sequence::is_dirty_main ()

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

Threadsafe

7.49.3.20 bool seq64::sequence::is_dirty_edit ()

Threadsafe

7.49.3.21 bool seq64::sequence::is_dirty_perf ()

Threadsafe

7.49.3.22 bool seq64::sequence::is_dirty_names ()

Threadsafe

7.49.3.23 void seq64::sequence::set_dirty_mp ()

Not threadsafe

7.49.3.24 void seq64::sequence::set_dirty ()

Threadsafe

7.49.3.25 void seq64::sequence::set_midi_channel (unsigned char a_ch)

Threadsafe

7.49.3.26 void seq64::sequence::print ()

Not threadsafe

7.49.3.27 void seq64::sequence::print_triggers ()

Not threadsafe

7.49.3.28 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

7.49.3.29 void seq64::sequence::set_orig_tick (long tick)

Threadsafe

7.49.3.30 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::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.

7.49.3.31 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
<      >
<      >

```

7.49.3.32 void seq64::sequence::split_trigger (long a_tick)

This is the public overload of split_trigger.

Threadsafe

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

Threadsafe

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

Threadsafe

7.49.3.35 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

<i>position</i>	The position to examine.
<i>start</i>	The destination for the starting tick (m_tick_start) of the matching trigger.
<i>end</i>	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.

7.49.3.36 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

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

7.49.3.37 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

<i>posstart</i>	The starting position to examine.
<i>posend</i>	The ending position to examine.
<i>status</i>	The desired status value.
<i>start</i>	The destination for the starting tick (<i>m_tick_start</i>) 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.

7.49.3.38 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

7.49.3.39 long seq64::sequence::selected_trigger_start ()

We guess this ends up selecting only one trigger, otherwise only the last selected on would set the result.

Threadsafe

7.49.3.40 long seq64::sequence::selected_trigger_end ()

Threadsafe

7.49.3.41 long seq64::sequence::get_max_trigger ()

*Threadsafe*7.49.3.42 void seq64::sequence::move_triggers (long *a_start_tick*, long *a_distance*, bool *a_direction*)*Threadsafe*7.49.3.43 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...
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

7.49.3.44 void seq64::sequence::clear_triggers ()

Threadsafe

7.49.3.45 void seq64::sequence::set_midi_bus (char mb)

Threadsafe

7.49.3.46 void seq64::sequence::set_master_midi_bus (mastermidibus * mmb)

Threadsafe

7.49.3.47 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

7.49.3.48 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

7.49.3.49 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.

7.49.3.50 int seq64::sequence::get_num_selected_notes ()

Threadsafe

7.49.3.51 `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

7.49.3.52 `void seq64::sequence::select_all ()`

Threadsafe

7.49.3.53 `void seq64::sequence::copy_selected ()`

Threadsafe

7.49.3.54 `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

7.49.3.55 `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

7.49.3.56 `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

7.49.3.57 `void seq64::sequence::stream_event (event * ev)`

Threadsafe

7.49.3.58 `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

$$d = \frac{df (t - ts) + ds (tf - t)}{tf - ts}$$

If this were an interpolation formula it would be:

$$d = ds + (df - ds) \frac{t - ts}{tf - ts}$$

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.
```

7.49.3.59 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

7.49.3.60 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

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

Threadsafe

7.49.3.62 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

7.49.3.63 void seq64::sequence::remove_marked ()

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

Threadsafe

7.49.3.64 void seq64::sequence::mark_selected ()

Threadsafe

7.49.3.65 void seq64::sequence::unpaint_all ()

Threadsafe

7.49.3.66 void seq64::sequence::unselect ()

Threadsafe

7.49.3.67 void seq64::sequence::verify_and_link ()

Threadsafe

7.49.3.68 void seq64::sequence::link_new ()

Threadsafe

7.49.3.69 void seq64::sequence::zero_markers ()

This function is used when the sequencer stops.

Threadsafe

7.49.3.70 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

7.49.3.71 void seq64::sequence::play_note_off (int a_note)

Threadsafe

7.49.3.72 void seq64::sequence::off_playing_notes ()

Threadsafe

7.49.3.73 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

7.49.3.74 `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.

7.49.3.75 `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.

7.49.3.76 `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.

7.49.3.77 `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

<i>c</i>	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).
<i>tracknumber</i>	Provides the track number. This number is masked into the track information.

7.49.3.78 `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.

7.49.3.79 `void seq64::sequence::put_event_on_bus (event * a_e) [private]`

Threadsafe

7.49.3.80 `void seq64::sequence::set_trigger_offset (long a_trigger_offset) [private]`

Threadsafe

7.49.3.81 `void seq64::sequence::split_trigger (trigger & trig, long a_split_tick) [private]`

The original trigger ends 1 tick before the *a_split_tick* parameter, and the new trigger starts at *a_split_tick* and ends where the original trigger ended.

This is the private overload of `split_trigger`.

Threadsafe

Parameters

<i>trig</i>	Provides the original trigger, and also holds the changes made to that trigger as it is shortened.
-------------	--

<code>a_split_tick</code>	The position just after where the original trigger will be truncated, and the new trigger begins.
---------------------------	---

7.49.3.82 void seq64::sequence::adjust_trigger_offsets_to_length (long a_new_len) [private]

```

|...|...|...|...|...|...|...|...
0123456789abcdef0123456789abcdef
[      ][      ][      ][      ][      ][
[  ][      ][  ][  ][  ][  ][  ][  ][  ][  ][
0  4      4  0 7 4 2 0      6  2
0  4      4  0 1 4 6 0      2  6 inverse offset

[      ][      ][      ][      ][      ][
[  ][      ][  ][  ][  ][  ][  ][  ][  ][  ][
0  c      4  0 f c a 8      e  a
0  4      c  0 1 4 6 8      2  6 inverse offset

[      ][      ][      ][      ][      ][
[  ][      ][  ][  ][  ][  ][  ][  ][  ][  ][
k   g f c a 8      [  ][  ][  ][
0  4      c   g h k m n      inverse offset

0123456789abcdefghijklmonpq
ponmlkjihgfedcba9876543210
0fedcba9876543210fedcba9876543210fedcba9876543210fedcba9876543210

```

Adjusts trigger offsets to the length of ???, for all triggers, and undo triggers.

Threadsafe

7.49.3.83 void seq64::sequence::remove (event_list::iterator i) [private]

If it's a note off, and that note is currently playing, then send a note off.

Not threadsafe

7.49.3.84 void seq64::sequence::remove (event * e) [private]

Finds the given event in m_events, and removes the first iterator matching that.

Not threadsafe

Todo Use find instead in `sequence::remove()`!

7.49.4 Field Documentation

7.49.4.1 mutex seq64::sequence::m_mutex [mutable], [private]

Made mutable for use in certain locked getter functions.

7.50 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.

7.50.1 Detailed Description

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

7.51 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.

Private Member Functions

- void [set_name](#) (const std::string &instname)
'Setter' function for member m_instrument_def.instrument
- void [copy_definitions](#) (const [user_instrument](#) &rhs)
Copies the array members from one instance of [user_instrument](#) to this one.

Private Attributes

- bool [m_is_valid](#)
Provides a validity flag, useful in returning a reference to a bogus object for internal error-check.
- int [m_controller_count](#)
Provides the actual number of non-default controllers actually set.
- [user_instrument_t](#) [m_instrument_def](#)
The instance of the structure that this class wraps.

7.51.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

7.51.2 Member Function Documentation

7.51.2.1 void user_instrument::set_defaults ()

Also invalidates the object.

7.51.2.2 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

<i>instrum</i>	Provides the destination instrument number. In order to support the legacy code, this index value must be less than c_max_instruments (64).
----------------	---

7.51.2.3 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

<i>instrum</i>	Provides the source instrument number. In order to support the legacy code, this index value must be less than c_max_instruments (64).
----------------	--

7.51.2.4 `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).

7.51.2.5 `const std::string & user_instrument::controller_name (int c) const`

Parameters

<code>c</code>	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.

7.51.2.6 `bool user_instrument::controller_active (int c) const`

Parameters

<code>c</code>	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.

7.51.2.7 `void user_instrument::set_controller (int c, const std::string & cname, bool isactive)`

Parameters

<code>c</code>	The index of the desired controller.
<code>cname</code>	The name of the controller to be set as the controller name.
<code>isactive</code>	A flag that indicates if the desired controller is active.

7.51.2.8 `void user_instrument::set_name (const std::string & instname) [private]`

If the name parameter is not empty, the validity flag is set to true, otherwise it is set to false. Too tricky?

7.51.2.9 `void user_instrument::copy_definitions (const user_instrument & rhs) [private]`

Does not include the validity flag.

7.51.3 Field Documentation

7.51.3.1 `bool user_instrument::m_is_valid [private]`

Callers should check this flag via the `is_valid()` accessor before using this object. This flag is set to true when any valid member assignment occurs via a public setter call. However, setting an empty name for the instrument member will render the object invalid.

7.51.3.2 int user_instrument::m_controller_count [private]

Often, the "user" configuration file has only a few out of the 128 assigned explicitly.

7.52 user_instrument_t Struct Reference

This structure corresponds to [user-instrument-N] definitions in the `~/ .seq24usr` or `~/ .config/sequencer64/sr rc` file.

7.53 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]

Private Member Functions

- void [set_name](#) (const std::string &name)
'Setter' function for member m_midi_bus_def.alias (name of alias) Also sets the validity flag according to the emptiness of the name parameter.
- void [copy_definitions](#) (const [user_midi_bus](#) &rhs)
Copies the member fields from one instance of [user_midi_bus](#) to this one.

Private Attributes

- bool [m_is_valid](#)
Provides a validity flag, useful in returning a reference to a bogus object for internal error-check.
- int [m_channel_count](#)
Provides the actual number of non-default buss channels actually set.
- [user_midi_bus_t](#) [m_midi_bus_def](#)
The instance of the structure that this class wraps.

7.53.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

7.53.2 Member Function Documentation

7.53.2.1 void user_midi_bus::set_defaults ()

Also invalidates the object. All 16 of the channels are set to GM_INSTRUMENT_FLAG (-1).

7.53.2.2 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

<i>buss</i>	Provides the destination buss number. In order to support the legacy code, this index value must be less than c_max_busses (32).
-------------	--

7.53.2.3 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

<i>buss</i>	Provides the destination buss number. In order to support the legacy code, this index value must be less than c_max_busses (32).
-------------	--

7.53.2.4 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.

7.53.2.5 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).

7.53.2.6 `int user_midi_bus::instrument (int channel) const`

Parameters

<i>channel</i>	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.

7.53.2.7 `void user_midi_bus::set_instrument (int channel, int instrum)`

Does not alter the validity flag, just checks it.

Parameters

<i>channel</i>	Provides the desired buss channel number.
<i>instrum</i>	Provides the instrument number to set that channel to.

7.53.2.8 `void user_midi_bus::copy_definitions (const user_midi_bus & rhs) [private]`

Does not include the validity flag.

7.53.3 Field Documentation

7.53.3.1 `bool user_midi_bus::m_is_valid [private]`

Callers should check this flag via the `is_valid()` accessor before using this object. This flag is set to true when any valid member assignment occurs via a public setter call.

7.53.3.2 `int user_midi_bus::m_channel_count [private]`

Often, the "user" configuration file has only a few out of the 16 assigned explicitly.

7.54 `user_midi_bus_t` Struct Reference

This structure corresponds to `[user-midi-bus-0]` definitions in the `~/ .seq24usr` ("user") file.

7.55 `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 &iname)
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`

Private Types

- typedef std::vector< `user_midi_bus` > `Busses`

Internal type for the container of `user_midi_bus` objects.

- typedef std::vector< `user_instrument` > `Instruments`

Internal type for the container of `user_instrument` objects.

Private Member Functions

- `user_midi_bus` & `private_bus` (int buss)

'Getter' function for member `m_midi_buses[index]` (internal function) If the index is out of range, then an invalid object is returned.

- `user_instrument` & `private_instrument` (int instrum)

'Getter' function for member `m_instruments[index]` If the index is out of range, then a invalid object is returned.

Private Attributes

- `Busses m_midi_buses`

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

- `Instruments m_instruments`

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

- int `m_mainwnd_rows`

Number of rows in the Patterns Panel.

- int `m_mainwnd_cols`

Number of columns in the Patterns Panel.

- int `m_seqs_in_set`

Number of patterns/sequences in the Patterns Panel, also known as a "set" or "screen set".

- int `m_gmute_tracks`

Number of group-mute tracks that can be support, which is `m_seqs_in_set` squared, or 1024.

- int `m_max_sets`

Maximum number of screen sets that can be supported.

- int `m_max_sequence`

The maximum number of patterns supported is given by the number of patterns supported in the panel (32) times the maximum number of sets (32), or 1024 patterns.

- int `m_text_x`

Constants for the mainwid class.

- int `m_seqchars_x`

Constants for the mainwid class.

- int `m_seqarea_x`

The `m_seqarea_x` and `m_seqarea_y` constants are derived from the width and heights of the default character set, and the number of characters in width, and the number of lines, in a pattern/sequence box.

- int `m_seqarea_seq_x`

Area of what? Doesn't look at all like it is based on the size of characters.

- int `m_mainwid_border`

These control sizes.

- int `m_control_height`

This constants seems to be created for a future purpose, perhaps to reserve space for a new bar on the mainwid pane.

- int `m_mainwid_x`

The width of the main pattern/sequence grid, in pixels.

7.55.1 Detailed Description

These settings will eventually be made part of the "user" settings file.

7.55.2 Member Typedef Documentation

7.55.2.1 `typedef std::vector<user_midi_bus> user_settings::Busses` `[private]`

Sorry about the "confusion" about "bus" versus "buss". See Google for arguments about it.

7.55.3 Member Function Documentation

7.55.3.1 `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!

7.55.3.2 `void user_settings::set_globals () const`

Should be called at initialization, and after settings are read from the "user" configuration file.

7.55.3.3 `void user_settings::get_globals ()`

Should be called before settings are written to the "user" configuration file.

7.55.3.4 `const user_midi_bus& user_settings::bus (int index)` `[inline]`

Cannot append the const specifier.

7.55.3.5 `const user_instrument& user_settings::instrument (int index)` `[inline]`

Cannot append the const specifier.

7.55.3.6 `int user_settings::bus_instrument (int buss, int channel)` `[inline]`

Todo Do this for controllers values and for `user_instrument` members.

7.55.3.7 void user_settings::mainwnd_rows (int value)

The default value is 4. Dependent values are recalculated after the assignment.

7.55.3.8 void user_settings::mainwnd_cols (int value)

The default value is 8. Dependent values are recalculated after the assignment.

7.55.3.9 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.

7.55.3.10 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.

7.55.3.11 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.

7.55.3.12 void user_settings::mainwid_border (int value)

The default value is 0. Dependent values are recalculated after the assignment.

7.55.3.13 void user_settings::mainwid_spacing (int value)

The default value is 2. Dependent values are recalculated after the assignment.

7.55.3.14 void user_settings::control_height (int value)

The default value is 0. Dependent values are recalculated after the assignment.

7.55.3.15 `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.

7.55.3.16 `user_midi_bus & user_settings::private_bus (int index) [private]`

This invalid object has an empty alias, and all the instrument numbers are -1.

7.55.3.17 `user_instrument & user_settings::private_instrument (int index) [private]`

This invalid object has an empty(), instrument name, false for all `controllers_active[]` values, and empty `controllers[]` string values.

7.55.4 Field Documentation

7.55.4.1 Busses `user_settings::m_midi_buses [private]`

Since this object is a vector, its size is adjustable.

7.55.4.2 Instruments `user_settings::m_instruments [private]`

The size is adjustable, and grows as objects are added.

7.55.4.3 `int user_settings::m_mainwnd_rows [private]`

The current value is 4, and if changed, many other values depend on it. Together with `m_mainwnd_cols`, this value fixes the patterns grid into a 4 x 8 set of patterns known as a "screen set".

7.55.4.4 `int user_settings::m_mainwnd_cols [private]`

The current value is 4, and probably won't change, since other values depend on it. Together with `m_mainwnd_rows`, this value fixes the patterns grid into a 4 x 8 set of patterns known as a "screen set".

7.55.4.5 `int user_settings::m_seqs_in_set [private]`

This value is 4 x 8 = 32 by default.

Warning

Currently part of the "rc" file and `rc_settings!`

7.55.4.6 `int user_settings::m_max_sets [private]`

Basically, that the number of times the Patterns Panel can be filled. 32 sets can be created.

7.55.4.7 `int user_settings::m_text_x` [private]

The `m_text_x` and `m_text_y` constants help define the "seqarea" size. It looks like these two values are the character width (x) and height (y) in pixels. Thus, these values would be dependent on the font chosen. But that, currently, is hard-wired. See the `m_font_6_12[]` array for the default font specification.

However, please not that font files are not used. Instead, the fonts are provided by two pixmaps in the `src/pixmap` directory: `font_b.xpm` (black lettering on a white background) and `font_w.xpm` (white lettering on a black background).

7.55.4.8 `int user_settings::m_seqchars_x` [private]

The `m_seqchars_x` and `m_seqchars_y` constants help define the "seqarea" size. These look like the number of characters per line and the number of lines of characters, in a pattern/sequence box.

7.55.4.9 `int user_settings::m_seqarea_x` [private]

Compare these two constants to `m_seqarea_seq_x(y)`, which was in `mainwid.h`, but is now in this file.

7.55.4.10 `int user_settings::m_seqarea_seq_x` [private]

These are used only in the `mainwid` module.

7.55.4.11 `int user_settings::m_mainwid_border` [private]

We'll try changing them and see what happens. Increasing these value spreads out the pattern grids a little bit and makes the Patterns panel slightly bigger. Seems like it would be useful to make these values user-configurable.

7.55.4.12 `int user_settings::m_control_height` [private]

But it is used only in this header file, to define `m_mainwid_y`, but doesn't add anything to that value.

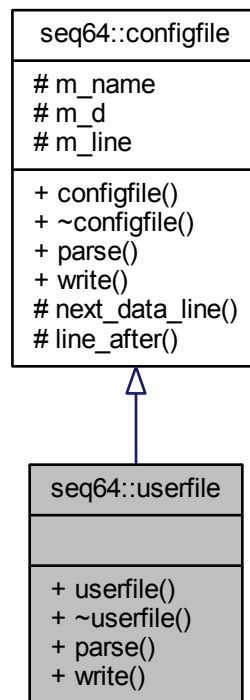
7.55.4.13 `int user_settings::m_mainwid_x` [private]

Affected by the `m_mainwid_border` and `m_mainwid_spacing` values.

7.56 `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

7.56.1 Member Function Documentation

7.56.1.1 bool seq64::userfile::parse (perform & a_perf) [virtual]

This function opens the file as a text file (line-oriented).

Parameters

<i>a_perf</i>	The performance object, currently unused.
---------------	---

Implements [seq64::configfile](#).

7.56.1.2 `bool seq64::userfile::write (const perform & a_perf) [virtual]`

Parameters

<i>a_perf</i>	The performance object, currently unused.
---------------	---

Implements [seq64::configfile](#).

Index

- ~keys_perform
 - seq64::keys_perform, [51](#)
- ~keys_perform_gtk2
 - seq64::keys_perform_gtk2, [55](#)
- ~perfedit
 - seq64::perfedit, [93](#)
- ~perform
 - seq64::perform, [102](#)
- ~seqmenu
 - seq64::seqmenu, [143](#)
- about_dialog
 - seq64::mainwnd, [72](#)
- add
 - seq64::event_list, [33](#)
- add_event
 - seq64::sequence, [159](#), [163](#)
- add_long
 - seq64::midi_container, [76](#)
- add_note
 - seq64::sequence, [163](#)
- add_sequence
 - seq64::perform, [103](#)
- add_trigger
 - seq64::sequence, [159](#)
- add_variable
 - seq64::midi_container, [76](#)
- adj_callback_ss
 - seq64::mainwnd, [72](#)
- adjust_trigger_offsets_to_length
 - seq64::sequence, [167](#)
- all_notes_off
 - seq64::perform, [105](#)
- append_sysex
 - seq64::event, [29](#)
- apply_length
 - seq64::seqedit, [131](#)
- BLACK
 - seq64::font, [36](#)
- BLACK_ON_YELLOW
 - seq64::font, [36](#)
- bus
 - user_settings, [177](#)
- bus_instrument
 - user_settings, [177](#)
- Busses
 - user_settings, [177](#)
- calculate_base_sizes
 - seq64::mainwid, [66](#)
- change_event_data_range
 - seq64::sequence, [163](#)
- channel_count
 - user_midi_bus, [172](#)
- channel_max
 - user_midi_bus, [172](#)
- CharList
 - seq64::midi_list, [78](#)
- clamp_track
 - seq64::perform, [110](#)
- clear_all
 - seq64::perform, [102](#)
- clear_sequence_triggers
 - seq64::perform, [103](#)
- clear_triggers
 - seq64::sequence, [162](#)
- click
 - seq64::click, [23](#)
- collapse
 - seq64::perfedit, [93](#)
- Color
 - seq64::font, [36](#)
- configfile
 - seq64::configfile, [25](#)
- control_height
 - user_settings, [178](#)
- controller_active
 - user_instrument, [170](#)
- controller_max
 - user_instrument, [169](#)
- controller_name
 - user_instrument, [170](#)
- convert_t
 - seq64::seqevent, [137](#)
- convert_tn
 - seq64::seqroll, [148](#)
- convert_x
 - seq64::perfroll, [114](#)
 - seq64::seqevent, [137](#)
- convert_xy
 - seq64::perfroll, [114](#)
- copy
 - seq64::perfedit, [93](#)
- copy_definitions
 - user_instrument, [170](#)
 - user_midi_bus, [173](#)
- copy_selected
 - seq64::sequence, [163](#)

- copy_triggers
 - seq64::perform, [103](#)
 - seq64::sequence, [161](#)
- count
 - seq64::event_list, [33](#)
- count_selected_events
 - seq64::event_list, [35](#)
- decrement_bpm
 - seq64::perform, [108](#)
- decrement_selected
 - seq64::sequence, [164](#)
- del_trigger
 - seq64::sequence, [160](#)
- delete_sequence
 - seq64::perform, [103](#)
- do_action
 - seq64::seqedit, [132](#)
- draw_background
 - seq64::seqevent, [136](#)
- draw_events_on_pixmap
 - seq64::seqroll, [148](#)
- draw_key
 - seq64::seqkeys, [141](#)
- draw_marker_on_sequence
 - seq64::mainwid, [64](#)
- draw_pixmap_on_window
 - seq64::seqevent, [136](#)
- draw_sequence_on
 - seq64::perfrill, [115](#)
- draw_sequence_on_pixmap
 - seq64::mainwid, [65](#)
- draw_sequence_pixmap_on_window
 - seq64::mainwid, [65](#)
- draw_sequences_on_pixmap
 - seq64::mainwid, [65](#)
- drop_event
 - seq64::seqevent, [137](#)
- dump_summary
 - user_settings, [178](#)
- e_deselect
 - seq64::sequence, [156](#)
- e_is_selected
 - seq64::sequence, [156](#)
- e_remove_one
 - seq64::sequence, [156](#)
- e_select
 - seq64::sequence, [156](#)
- e_select_one
 - seq64::sequence, [156](#)
- e_toggle_selection
 - seq64::sequence, [156](#)
- e_would_select
 - seq64::sequence, [156](#)
- event_count
 - seq64::sequence, [156](#)
- event_key
 - seq64::event_list::event_key, [31](#)
- event_list
 - seq64::event_list, [33](#)
- events
 - seq64::keybindentry, [48](#)
- expand
 - seq64::perfdit, [93](#)
- file_import_dialog
 - seq64::mainwnd, [71](#)
- fill
 - seq64::midi_container, [75](#)
- fill_container
 - seq64::sequence, [166](#)
- get
 - seq64::midi_container, [75](#)
 - seq64::midi_list, [78](#)
 - seq64::midi_vector, [80](#)
- get_bw
 - seq64::sequence, [157](#)
- get_data
 - seq64::event, [29](#)
- get_global
 - user_instrument, [169](#)
 - user_midi_bus, [172](#)
- get_globals
 - user_settings, [177](#)
- get_keys
 - seq64::keys_perform, [52](#)
- get_max_trigger
 - seq64::perform, [108](#)
 - seq64::sequence, [161](#)
- get_measures
 - seq64::seqedit, [131](#)
- get_midi_control_off
 - seq64::perform, [104](#)
- get_midi_control_on
 - seq64::perform, [104](#)
- get_midi_control_toggle
 - seq64::perform, [103](#)
- get_next_event
 - seq64::sequence, [166](#)
- get_next_note_event
 - seq64::sequence, [165](#)
- get_num_selected_events
 - seq64::sequence, [162](#)
- get_num_selected_notes
 - seq64::sequence, [162](#)
- get_rank
 - seq64::event, [29](#)
- get_screen_set_notepad
 - seq64::perform, [104](#)
- get_sequence
 - seq64::perform, [106](#)
- groups
 - seq64::keybindentry, [48](#)
- grow_selected
 - seq64::sequence, [164](#)
- grow_trigger

- seq64::sequence, 159
- gui_assistant
 - seq64::gui_assistant, 39
- gui_palette_gtk2
 - seq64::gui_palette_gtk2, 43
- gui_window_gtk2
 - seq64::gui_window_gtk2, 45
- handle_config
 - seq64::lash, 59
- handle_event
 - seq64::lash, 59
- home_config_directory
 - rc_settings, 120
- idle_progress
 - seq64::maintime, 61
- idle_redraw
 - seq64::seqdata, 126
 - seq64::seqevent, 136
- increment_bpm
 - seq64::perform, 108
- increment_selected
 - seq64::sequence, 164
- init
 - seq64::font, 36
 - seq64::jack_assistant, 46
 - seq64::lash, 59
 - seq64::perform, 102
- init_before_show
 - seq64::perfeddit, 93
 - seq64::perfroll, 114
- init_jack
 - seq64::perform, 103
- inner_start
 - seq64::perform, 110
- install_sequence
 - seq64::perform, 109
- instrument
 - user_midi_bus, 172
 - user_settings, 177
- intersectEvents
 - seq64::sequence, 160
- intersectNotes
 - seq64::sequence, 160
- intersectTriggers
 - seq64::sequence, 160
- is_active
 - seq64::perform, 105
- is_dirty_edit
 - seq64::perform, 106
 - seq64::sequence, 158
- is_dirty_main
 - seq64::perform, 106
 - seq64::sequence, 158
- is_dirty_names
 - seq64::perform, 106
 - seq64::sequence, 158
- is_dirty_perf
 - seq64::perform, 106
 - seq64::sequence, 158
- is_letter
 - seq64::keystroke, 57
- is_midi_control_valid
 - seq64::perform, 109
- is_mseq_valid
 - seq64::perform, 109
- is_screenset_valid
 - seq64::perform, 109
- is_seq_valid
 - seq64::perform, 109
- jack_assistant
 - seq64::jack_assistant, 46
- jack_shutdown
 - seq64::jack_assistant, 47
- jack_sync_callback
 - seq64::jack_assistant, 47
 - seq64::perform, 110
- jack_timebase_callback
 - seq64::jack_assistant, 47
- key_name
 - seq64::keys_perform, 52
 - seq64::keys_perform_gtk2, 55
- keybindentry
 - seq64::keybindentry, 48
- keystroke
 - seq64::keystroke, 56, 57
- lash
 - seq64::lash, 58
- launch_input_thread
 - seq64::perform, 102
- launch_output_thread
 - seq64::perform, 102
- line_after
 - seq64::configfile, 25
- link_new
 - seq64::event_list, 34
 - seq64::sequence, 165
- location
 - seq64::keybindentry, 48
- m_b_on_y_pixmap
 - seq64::font, 37
- m_black_pixmap
 - seq64::font, 36
- m_button
 - seq64::click, 23
- m_channel_count
 - user_midi_bus, 173
- m_char_list
 - seq64::midifile, 87
- m_clip_mask
 - seq64::font, 37
- m_control_height
 - user_settings, 180

- m_controller_count
 - user_instrument, 170
- m_data
 - seq64::event, 30
 - seq64::midifile, 87
- m_drop_x
 - seq64::gui_drawingarea_gtk2, 42
- m_has_link
 - seq64::event, 30
- m_instruments
 - user_settings, 179
- m_is_press
 - seq64::keystroke, 57
- m_is_valid
 - user_instrument, 170
 - user_midi_bus, 173
- m_key
 - seq64::keybindentry, 49
 - seq64::keystroke, 57
- m_key_bpm_up
 - seq64::keys_perform, 53
- m_line
 - seq64::configfile, 25
- m_main_wid
 - seq64::mainwnd, 73
- m_mainperf
 - seq64::gui_drawingarea_gtk2, 42
- m_mainwid_border
 - user_settings, 180
- m_mainwid_x
 - user_settings, 180
- m_mainwnd_cols
 - user_settings, 179
- m_mainwnd_rows
 - user_settings, 179
- m_max_sets
 - user_settings, 179
- m_midi_buses
 - user_settings, 179
- m_modifier
 - seq64::click, 23
 - seq64::keystroke, 57
- m_mutex
 - seq64::sequence, 167
- m_new_format
 - seq64::midifile, 87
- m_notebook
 - seq64::options, 88
- m_pixmap
 - seq64::font, 36
- m_playback_mode
 - seq64::perform, 110
- m_pos
 - seq64::midifile, 87
- m_rank
 - seq64::event_list::event_key, 31
- m_seqarea_seq_x
 - user_settings, 180
- m_seqarea_x
 - user_settings, 180
- m_seqchars_x
 - user_settings, 180
- m_seqedit
 - seq64::seqmenu, 144
- m_seqs_in_set
 - user_settings, 179
- m_sigpipe
 - seq64::mainwnd, 73
- m_spinbutton_load_offset
 - seq64::mainwnd, 73
- m_status
 - seq64::event, 30
- m_sysex
 - seq64::event, 30
- m_text_x
 - user_settings, 179
- m_timestamp
 - seq64::event_list::event_key, 31
- m_white_pixmap
 - seq64::font, 37
- m_window_x
 - seq64::gui_drawingarea_gtk2, 42
 - seq64::gui_window_gtk2, 45
- m_x
 - seq64::click, 23
- m_y
 - seq64::click, 23
- m_y_on_b_pixmap
 - seq64::font, 37
- maintime
 - seq64::maintime, 61
- mainwid
 - seq64::mainwid, 64
- mainwid_border
 - user_settings, 178
- mainwid_spacing
 - user_settings, 178
- mainwnd
 - seq64::mainwnd, 71
- mainwnd_cols
 - user_settings, 178
- mainwnd_key_event
 - seq64::perform, 108
- mainwnd_rows
 - user_settings, 177
- make_directory
 - rc_settings, 120
- mark_out_of_range
 - seq64::event_list, 35
- mark_selected
 - seq64::sequence, 165
- max_sets
 - user_settings, 178
- mc_min_zoom
 - seq64::seqedit, 133
- merge

- seq64::event_list, 34
- midifile
 - seq64::midifile, 82
- mod_timestamp
 - seq64::event, 28
- move_selected_triggers_to
 - seq64::sequence, 161
- move_triggers
 - seq64::perform, 103
 - seq64::sequence, 161
- name_change_callback
 - seq64::seqedit, 132
- new_sequence
 - seq64::perform, 106
- next_data_line
 - seq64::configfile, 25
- off_playing_notes
 - seq64::sequence, 165
- off_queued
 - seq64::sequence, 157
- on_button_press_event
 - seq64::Seq24PerfInput, 121
 - seq64::Seq24SeqEventInput, 122
 - seq64::mainwid, 66
 - seq64::perfroll, 115
 - seq64::seqevent, 137
 - seq64::seqkeys, 141
 - seq64::seqtime, 150
- on_button_release_event
 - seq64::Seq24PerfInput, 121
 - seq64::mainwid, 66
 - seq64::perfroll, 115
 - seq64::seqevent, 137
 - seq64::seqkeys, 141
 - seq64::seqtime, 150
- on_delete_event
 - seq64::mainwnd, 73
 - seq64::seqedit, 132
- on_expose_event
 - seq64::mainwid, 66
 - seq64::perfnames, 96
 - seq64::perftime, 117
- on_focus_in_event
 - seq64::mainwid, 67
- on_focus_out_event
 - seq64::mainwid, 67
- on_grouplearnchange
 - seq64::mainwnd, 73
- on_key_press_event
 - seq64::keybindentry, 49
 - seq64::mainwnd, 73
 - seq64::perfroll, 115
 - seq64::seqevent, 138
 - seq64::seqroll, 148
- on_key_release_event
 - seq64::mainwnd, 73
- on_motion_notify_event
 - seq64::mainwid, 67
 - seq64::seqdata, 126
 - seq64::seqevent, 138
- on_realize
 - seq64::gui_drawingarea_gtk2, 41
 - seq64::maintime, 61
 - seq64::mainwid, 66
 - seq64::perfnames, 96
 - seq64::perfroll, 115
 - seq64::perftime, 117
 - seq64::seqdata, 126
 - seq64::seqevent, 137
 - seq64::seqkeys, 141
- on_scroll_event
 - seq64::seqdata, 127
 - seq64::seqroll, 148
- on_size_allocate
 - seq64::perfnames, 96
- open_file
 - seq64::mainwnd, 71
- open_performance_edit
 - seq64::mainwnd, 72
- operator<
 - seq64::event, 28
 - seq64::event_list::event_key, 31
- operator=
 - seq64::click, 23
 - seq64::event_list, 33
 - seq64::keystroke, 57
 - seq64::sequence, 156
- output
 - seq64::jack_assistant, 46
- output_func
 - seq64::perform, 107
- parse
 - seq64::midifile, 82
 - seq64::optionsfile, 89
 - seq64::userfile, 181
- parse_prop_header
 - seq64::midifile, 83
- parse_proprietary_track
 - seq64::midifile, 84
- paste_selected
 - seq64::sequence, 163
- perfedit
 - seq64::perfedit, 93
- perfnames
 - seq64::perfnames, 96
- perform
 - seq64::perform, 102
- perfroll_key_event
 - seq64::perform, 108
- perftime
 - seq64::perftime, 117
- play
 - seq64::perform, 107
 - seq64::sequence, 158
- play_note_off

- seq64::sequence, 165
- play_note_on
 - seq64::sequence, 165
- pop_redo
 - seq64::sequence, 156
- pop_undo
 - seq64::sequence, 156
- popup_event_menu
 - seq64::seqedit, 132
- popup_midibus_menu
 - seq64::seqedit, 132
- popup_sequence_menu
 - seq64::seqedit, 132
- popup_tool_menu
 - seq64::seqedit, 132
- position
 - seq64::jack_assistant, 46
 - seq64::midi_container, 75
- position_jack
 - seq64::perform, 105
- ppqn
 - seq64::mainwnd, 71
 - seq64::midifile, 83
- print
 - seq64::sequence, 158
- print_triggers
 - seq64::sequence, 158
- private_bus
 - user_settings, 179
- private_instrument
 - user_settings, 179
- process_events
 - seq64::lash, 59
- prop_item_size
 - seq64::midifile, 87
- push_trigger_undo
 - seq64::sequence, 156
- push_undo
 - seq64::sequence, 156
- put
 - seq64::midi_container, 75
 - seq64::midi_list, 78
 - seq64::midi_vector, 80
- put_event_on_bus
 - seq64::sequence, 166
- rc_settings, 118
 - home_config_directory, 120
 - make_directory, 120
- read_long
 - seq64::midifile, 84
- read_short
 - seq64::midifile, 84
- read_varinum
 - seq64::midifile, 84
- redraw
 - seq64::mainwid, 66
 - seq64::seqdata, 126
- remove
 - seq64::sequence, 167
- remove_marked
 - seq64::sequence, 164
- render_string_on_drawable
 - seq64::font, 36
- reset
 - seq64::seqdata, 126
 - seq64::seqroll, 147
- reset_draw_marker
 - seq64::sequence, 165
- reset_sequences
 - seq64::perform, 107
- save_file
 - seq64::mainwnd, 72
- select_action_e
 - seq64::sequence, 156
- select_all
 - seq64::sequence, 163
- select_events
 - seq64::sequence, 162
- select_mute_group
 - seq64::perform, 105
- select_note_events
 - seq64::sequence, 162
- selected_trigger_end
 - seq64::sequence, 161
- selected_trigger_start
 - seq64::sequence, 161
- seq64::AbstractPerfInput, 21
- seq64::Seq24PerfInput, 120
 - on_button_press_event, 121
 - on_button_release_event, 121
 - set_adding, 122
- seq64::Seq24SeqEventInput, 122
 - on_button_press_event, 122
 - set_adding, 122
- seq64::Seq24SeqRollInput, 122
 - set_adding, 123
- seq64::click, 21
 - click, 23
 - m_button, 23
 - m_modifier, 23
 - m_x, 23
 - m_y, 23
 - operator=, 23
- seq64::configfile, 24
 - configfile, 25
 - line_after, 25
 - m_line, 25
 - next_data_line, 25
- seq64::event, 25
 - append_sysex, 29
 - get_data, 29
 - get_rank, 29
 - m_data, 30
 - m_has_link, 30
 - m_status, 30
 - m_sysex, 30

- mod_timestamp, 28
- operator<, 28
- set_data, 29
- set_status, 29
- seq64::event_list, 31
 - add, 33
 - count, 33
 - count_selected_events, 35
 - event_list, 33
 - link_new, 34
 - mark_out_of_range, 35
 - merge, 34
 - operator=, 33
 - verify_and_link, 34
- seq64::event_list::event_key, 30
 - event_key, 31
 - m_rank, 31
 - m_timestamp, 31
 - operator<, 31
- seq64::font, 35
 - BLACK, 36
 - BLACK_ON_YELLOW, 36
 - Color, 36
 - init, 36
 - m_b_on_y_pixmap, 37
 - m_black_pixmap, 36
 - m_clip_mask, 37
 - m_pixmap, 36
 - m_white_pixmap, 37
 - m_y_on_b_pixmap, 37
 - render_string_on_drawable, 36
 - WHITE, 36
 - YELLOW_ON_BLACK, 36
- seq64::gui_assistant, 37
 - gui_assistant, 39
- seq64::gui_assistant_gtk2, 39
 - sm_internal_keys, 40
- seq64::gui_drawingarea_gtk2, 40
 - m_drop_x, 42
 - m_mainperf, 42
 - m_window_x, 42
 - on_realize, 41
- seq64::gui_drawingarea_gtk2::rect, 120
- seq64::gui_palette_gtk2, 42
 - gui_palette_gtk2, 43
- seq64::gui_window_gtk2, 43
 - gui_window_gtk2, 45
 - m_window_x, 45
- seq64::jack_assistant, 45
 - init, 46
 - jack_assistant, 46
 - jack_shutdown, 47
 - jack_sync_callback, 47
 - jack_timebase_callback, 47
 - output, 46
 - position, 46
- seq64::jack_scratchpad, 47
- seq64::keybindentry, 47
 - events, 48
 - groups, 48
 - keybindentry, 48
 - location, 48
 - m_key, 49
 - on_key_press_event, 49
 - set, 49
 - type, 48
- seq64::keys_perform, 49
 - ~keys_perform, 51
 - get_keys, 52
 - key_name, 52
 - m_key_bpm_up, 53
 - set_all_key_events, 52
 - set_all_key_groups, 52
 - set_key_event, 52
 - set_key_group, 52
 - set_keys, 51
 - show_ui_sequence_key, 52
- seq64::keys_perform_gtk2, 53
 - ~keys_perform_gtk2, 55
 - key_name, 55
 - set_all_key_events, 55
 - set_all_key_groups, 55
- seq64::keys_perform_transfer, 55
- seq64::keystroke, 55
 - is_letter, 57
 - keystroke, 56, 57
 - m_is_press, 57
 - m_key, 57
 - m_modifier, 57
 - operator=, 57
- seq64::lash, 58
 - handle_config, 59
 - handle_event, 59
 - init, 59
 - lash, 58
 - process_events, 59
 - set_alsa_client_id, 59
- seq64::maintime, 59
 - idle_progress, 61
 - maintime, 61
 - on_realize, 61
- seq64::mainwid, 61
 - calculate_base_sizes, 66
 - draw_marker_on_sequence, 64
 - draw_sequence_on_pixmap, 65
 - draw_sequence_pixmap_on_window, 65
 - draw_sequences_on_pixmap, 65
 - mainwid, 64
 - on_button_press_event, 66
 - on_button_release_event, 66
 - on_expose_event, 66
 - on_focus_in_event, 67
 - on_focus_out_event, 67
 - on_motion_notify_event, 67
 - on_realize, 66
 - redraw, 66

- seq_from_xy, 65
- set_screenset, 64
- timeout, 65
- update_markers, 64
- update_sequence_on_window, 64
- valid_sequence, 64
- seq64::mainwnd, 67
 - about_dialog, 72
 - adj_callback_ss, 72
 - file_import_dialog, 71
 - m_main_wid, 73
 - m_sigpipe, 73
 - m_spinbutton_load_offset, 73
 - mainwnd, 71
 - on_delete_event, 73
 - on_grouplearchange, 73
 - on_key_press_event, 73
 - on_key_release_event, 73
 - open_file, 71
 - open_performance_edit, 72
 - ppqn, 71
 - save_file, 72
 - signal_action, 72
 - timer_callback, 72
 - update_window_title, 72
- seq64::midi_container, 74
 - add_long, 76
 - add_variable, 76
 - fill, 75
 - get, 75
 - position, 75
 - put, 75
- seq64::midi_list, 76
 - CharList, 78
 - get, 78
 - put, 78
- seq64::midi_vector, 78
 - get, 80
 - put, 80
- seq64::midifile, 80
 - m_char_list, 87
 - m_data, 87
 - m_new_format, 87
 - m_pos, 87
 - midifile, 82
 - parse, 82
 - parse_prop_header, 83
 - parse_proprietary_track, 84
 - ppqn, 83
 - prop_item_size, 87
 - read_long, 84
 - read_short, 84
 - read_varinum, 84
 - seq_number_size, 87
 - varinum_size, 86
 - write, 83
 - write_byte, 85
 - write_long, 84
 - write_prop_header, 85
 - write_proprietary_track, 86
 - write_seq_number, 85
 - write_short, 85
 - write_track_name, 85
 - write_varinum, 85
- seq64::options, 87
 - m_notebook, 88
- seq64::optionsfile, 88
 - parse, 89
 - write, 90
- seq64::perfedit, 90
 - ~perfedit, 93
 - collapse, 93
 - copy, 93
 - expand, 93
 - init_before_show, 93
 - perfedit, 93
 - set_guides, 93
 - start_playing, 94
 - timeout, 94
 - undo, 93
- seq64::perfnames, 94
 - on_expose_event, 96
 - on_realize, 96
 - on_size_allocate, 96
 - perfnames, 96
- seq64::perform, 97
 - ~perform, 102
 - add_sequence, 103
 - all_notes_off, 105
 - clamp_track, 110
 - clear_all, 102
 - clear_sequence_triggers, 103
 - copy_triggers, 103
 - decrement_bpm, 108
 - delete_sequence, 103
 - get_max_trigger, 108
 - get_midi_control_off, 104
 - get_midi_control_on, 104
 - get_midi_control_toggle, 103
 - get_screen_set_notepad, 104
 - get_sequence, 106
 - increment_bpm, 108
 - init, 102
 - init_jack, 103
 - inner_start, 110
 - install_sequence, 109
 - is_active, 105
 - is_dirty_edit, 106
 - is_dirty_main, 106
 - is_dirty_names, 106
 - is_dirty_perf, 106
 - is_midi_control_valid, 109
 - is_mseq_valid, 109
 - is_screenset_valid, 109
 - is_seq_valid, 109
 - jack_sync_callback, 110

- launch_input_thread, 102
- launch_output_thread, 102
- m_playback_mode, 110
- mainwnd_key_event, 108
- move_triggers, 103
- new_sequence, 106
- output_func, 107
- perform, 102
- perfroll_key_event, 108
- play, 107
- position_jack, 105
- reset_sequences, 107
- select_mute_group, 105
- sequence_count, 102
- set_active, 105
- set_bpm, 107
- set_input_bus, 108
- set_key_event, 110
- set_key_group, 110
- set_offset, 108
- set_orig_ticks, 107
- set_playing_screensets, 104
- set_screen_set_notepad, 104
- set_screensets, 104
- set_sequence_control_status, 107
- set_was_active, 105
- show_ui_sequence_key, 108
- start, 105
- start_playing, 108
- stop, 105
- unset_mode_group_learn, 104
- unset_sequence_control_status, 107
- seq64::performcallback, 110
- seq64::perfroll, 111
 - convert_x, 114
 - convert_xy, 114
 - draw_sequence_on, 115
 - init_before_show, 114
 - on_button_press_event, 115
 - on_button_release_event, 115
 - on_key_press_event, 115
 - on_realize, 115
 - snap_x, 114
 - start_playing, 115
 - stop_playing, 115
 - update_sizes, 114
- seq64::perftime, 115
 - on_expose_event, 117
 - on_realize, 117
 - perftime, 117
- seq64::rect, 120
- seq64::seqdata, 123
 - idle_redraw, 126
 - on_motion_notify_event, 126
 - on_realize, 126
 - on_scroll_event, 127
 - redraw, 126
 - reset, 126
 - seqdata, 126
 - set_zoom, 126
 - update_sizes, 126
 - xy_to_rect, 126
- seq64::seqedit, 127
 - apply_length, 131
 - do_action, 132
 - get_measures, 131
 - mc_min_zoom, 133
 - name_change_callback, 132
 - on_delete_event, 132
 - popup_event_menu, 132
 - popup_midibus_menu, 132
 - popup_sequence_menu, 132
 - popup_tool_menu, 132
 - seqedit, 131
 - set_background_sequence, 132
 - set_data_type, 132
 - set_key, 132
 - set_note_length, 131
 - set_scale, 131
 - set_snap, 131
 - set_zoom, 131
- seq64::seqevent, 133
 - convert_t, 137
 - convert_x, 137
 - draw_background, 136
 - draw_pixmap_on_window, 136
 - drop_event, 137
 - idle_redraw, 136
 - on_button_press_event, 137
 - on_button_release_event, 137
 - on_key_press_event, 138
 - on_motion_notify_event, 138
 - on_realize, 137
 - set_data_type, 136
 - set_snap, 136
 - snap_x, 137
 - start_paste, 137
 - update_sizes, 136
 - x_to_w, 137
- seq64::seqkeys, 138
 - draw_key, 141
 - on_button_press_event, 141
 - on_button_release_event, 141
 - on_realize, 141
 - set_hint_state, 140
- seq64::seqmenu, 141
 - ~seqmenu, 143
 - m_seqedit, 144
 - seq_clear_perf, 144
 - seq_copy, 144
 - seq_cut, 144
 - seq_edit, 144
 - seq_paste, 144
 - seqmenu, 143
- seq64::seqroll, 144
 - convert_tn, 148

- draw_events_on_pixmap, 148
- on_key_press_event, 148
- on_scroll_event, 148
- reset, 147
- set_background_sequence, 148
- set_data_type, 147
- snap_x, 148
- seq64::seqtime, 148
 - on_button_press_event, 150
 - on_button_release_event, 150
- seq64::sequence, 150
 - add_event, 159, 163
 - add_note, 163
 - add_trigger, 159
 - adjust_trigger_offsets_to_length, 167
 - change_event_data_range, 163
 - clear_triggers, 162
 - copy_selected, 163
 - copy_triggers, 161
 - decrement_selected, 164
 - del_trigger, 160
 - e_deselect, 156
 - e_is_selected, 156
 - e_remove_one, 156
 - e_select, 156
 - e_select_one, 156
 - e_toggle_selection, 156
 - e_would_select, 156
 - event_count, 156
 - fill_container, 166
 - get_bw, 157
 - get_max_trigger, 161
 - get_next_event, 166
 - get_next_note_event, 165
 - get_num_selected_events, 162
 - get_num_selected_notes, 162
 - grow_selected, 164
 - grow_trigger, 159
 - increment_selected, 164
 - intersectEvents, 160
 - intersectNotes, 160
 - intersectTriggers, 160
 - is_dirty_edit, 158
 - is_dirty_main, 158
 - is_dirty_names, 158
 - is_dirty_perf, 158
 - link_new, 165
 - m_mutex, 167
 - mark_selected, 165
 - move_selected_triggers_to, 161
 - move_triggers, 161
 - off_playing_notes, 165
 - off_queued, 157
 - operator=, 156
 - paste_selected, 163
 - play, 158
 - play_note_off, 165
 - play_note_on, 165
 - pop_redo, 156
 - pop_undo, 156
 - print, 158
 - print_triggers, 158
 - push_trigger_undo, 156
 - push_undo, 156
 - put_event_on_bus, 166
 - remove, 167
 - remove_marked, 164
 - reset_draw_marker, 165
 - select_action_e, 156
 - select_all, 163
 - select_events, 162
 - select_note_events, 162
 - selected_trigger_end, 161
 - selected_trigger_start, 161
 - set_bpm, 157
 - set_bw, 157
 - set_dirty, 158
 - set_dirty_mp, 158
 - set_length, 157
 - set_master_midi_bus, 162
 - set_midi_bus, 162
 - set_midi_channel, 158
 - set_orig_tick, 159
 - set_playing, 157
 - set_quantized_rec, 157
 - set_rec_vol, 157
 - set_recording, 157
 - set_snap_tick, 157
 - set_thru, 158
 - set_trigger_offset, 166
 - split_trigger, 159, 166
 - stream_event, 163
 - stretch_selected, 164
 - toggle_queued, 157
 - transpose_notes, 166
 - unpaint_all, 165
 - unselect, 165
 - verify_and_link, 165
 - zero_markers, 165
- seq64::trigger, 167
- seq64::userfile, 180
 - parse, 181
 - write, 182
- seq_clear_perf
 - seq64::seqmenu, 144
- seq_copy
 - seq64::seqmenu, 144
- seq_cut
 - seq64::seqmenu, 144
- seq_edit
 - seq64::seqmenu, 144
- seq_from_xy
 - seq64::mainwid, 65
- seq_number_size
 - seq64::midifile, 87
- seq_paste

- seq64::seqmenu, 144
- seqdata
 - seq64::seqdata, 126
- seqedit
 - seq64::seqedit, 131
- seqmenu
 - seq64::seqmenu, 143
- sequence_count
 - seq64::perform, 102
- set
 - seq64::keybindentry, 49
- set_active
 - seq64::perform, 105
- set_adding
 - seq64::Seq24PerfInput, 122
 - seq64::Seq24SeqEventInput, 122
 - seq64::Seq24SeqRollInput, 123
- set_all_key_events
 - seq64::keys_perform, 52
 - seq64::keys_perform_gtk2, 55
- set_all_key_groups
 - seq64::keys_perform, 52
 - seq64::keys_perform_gtk2, 55
- set_alsa_client_id
 - seq64::lash, 59
- set_background_sequence
 - seq64::seqedit, 132
 - seq64::seqroll, 148
- set_bpm
 - seq64::perform, 107
 - seq64::sequence, 157
- set_bw
 - seq64::sequence, 157
- set_controller
 - user_instrument, 170
- set_data
 - seq64::event, 29
- set_data_type
 - seq64::seqedit, 132
 - seq64::seqevent, 136
 - seq64::seqroll, 147
- set_defaults
 - user_instrument, 169
 - user_midi_bus, 172
 - user_settings, 177
- set_dirty
 - seq64::sequence, 158
- set_dirty_mp
 - seq64::sequence, 158
- set_global
 - user_instrument, 169
 - user_midi_bus, 172
- set_globals
 - user_settings, 177
- set_guides
 - seq64::perfedit, 93
- set_hint_state
 - seq64::seqkeys, 140
- set_input_bus
 - seq64::perform, 108
- set_instrument
 - user_midi_bus, 173
- set_key
 - seq64::seqedit, 132
- set_key_event
 - seq64::keys_perform, 52
 - seq64::perform, 110
- set_key_group
 - seq64::keys_perform, 52
 - seq64::perform, 110
- set_keys
 - seq64::keys_perform, 51
- set_length
 - seq64::sequence, 157
- set_master_midi_bus
 - seq64::sequence, 162
- set_midi_bus
 - seq64::sequence, 162
- set_midi_channel
 - seq64::sequence, 158
- set_name
 - user_instrument, 170
- set_note_length
 - seq64::seqedit, 131
- set_offset
 - seq64::perform, 108
- set_orig_tick
 - seq64::sequence, 159
- set_orig_ticks
 - seq64::perform, 107
- set_playing
 - seq64::sequence, 157
- set_playing_screensets
 - seq64::perform, 104
- set_quantized_rec
 - seq64::sequence, 157
- set_rec_vol
 - seq64::sequence, 157
- set_recording
 - seq64::sequence, 157
- set_scale
 - seq64::seqedit, 131
- set_screen_set_notepad
 - seq64::perform, 104
- set_screensets
 - seq64::mainwid, 64
 - seq64::perform, 104
- set_sequence_control_status
 - seq64::perform, 107
- set_snap
 - seq64::seqedit, 131
 - seq64::seqevent, 136
- set_snap_tick
 - seq64::sequence, 157
- set_status
 - seq64::event, 29

- set_thru
 - seq64::sequence, 158
- set_trigger_offset
 - seq64::sequence, 166
- set_was_active
 - seq64::perform, 105
- set_zoom
 - seq64::seqdata, 126
 - seq64::seqedit, 131
- show_ui_sequence_key
 - seq64::keys_perform, 52
 - seq64::perform, 108
- signal_action
 - seq64::mainwnd, 72
- sm_internal_keys
 - seq64::gui_assistant_gtk2, 40
- snap_x
 - seq64::perfroll, 114
 - seq64::seqevent, 137
 - seq64::seqroll, 148
- split_trigger
 - seq64::sequence, 159, 166
- start
 - seq64::perform, 105
- start_paste
 - seq64::seqevent, 137
- start_playing
 - seq64::perfedit, 94
 - seq64::perform, 108
 - seq64::perfroll, 115
- stop
 - seq64::perform, 105
- stop_playing
 - seq64::perfroll, 115
- stream_event
 - seq64::sequence, 163
- stretch_selected
 - seq64::sequence, 164
- text_x
 - user_settings, 178
- text_y
 - user_settings, 178
- timeout
 - seq64::mainwid, 65
 - seq64::perfedit, 94
- timer_callback
 - seq64::mainwnd, 72
- toggle_queued
 - seq64::sequence, 157
- transpose_notes
 - seq64::sequence, 166
- type
 - seq64::keybindentry, 48
- undo
 - seq64::perfedit, 93
- unpaint_all
 - seq64::sequence, 165
- unselect
 - seq64::sequence, 165
- unset_mode_group_learn
 - seq64::perform, 104
- unset_sequence_control_status
 - seq64::perform, 107
- update_markers
 - seq64::mainwid, 64
- update_sequence_on_window
 - seq64::mainwid, 64
- update_sizes
 - seq64::perfroll, 114
 - seq64::seqdata, 126
 - seq64::seqevent, 136
- update_window_title
 - seq64::mainwnd, 72
- user_instrument, 168
 - controller_active, 170
 - controller_max, 169
 - controller_name, 170
 - copy_definitions, 170
 - get_global, 169
 - m_controller_count, 170
 - m_is_valid, 170
 - set_controller, 170
 - set_defaults, 169
 - set_global, 169
 - set_name, 170
- user_instrument_t, 171
- user_midi_bus, 171
 - channel_count, 172
 - channel_max, 172
 - copy_definitions, 173
 - get_global, 172
 - instrument, 172
 - m_channel_count, 173
 - m_is_valid, 173
 - set_defaults, 172
 - set_global, 172
 - set_instrument, 173
- user_midi_bus_t, 173
- user_settings, 173
 - bus, 177
 - bus_instrument, 177
 - Busses, 177
 - control_height, 178
 - dump_summary, 178
 - get_globals, 177
 - instrument, 177
 - m_control_height, 180
 - m_instruments, 179
 - m_mainwid_border, 180
 - m_mainwid_x, 180
 - m_mainwnd_cols, 179
 - m_mainwnd_rows, 179
 - m_max_sets, 179
 - m_midi_buses, 179
 - m_seqarea_seq_x, 180

- m_seqarea_x, [180](#)
- m_seqchars_x, [180](#)
- m_seqs_in_set, [179](#)
- m_text_x, [179](#)
- mainwid_border, [178](#)
- mainwid_spacing, [178](#)
- mainwnd_cols, [178](#)
- mainwnd_rows, [177](#)
- max_sets, [178](#)
- private_bus, [179](#)
- private_instrument, [179](#)
- set_defaults, [177](#)
- set_globals, [177](#)
- text_x, [178](#)
- text_y, [178](#)

valid_sequence

- seq64::mainwid, [64](#)

varinum_size

- seq64::midifile, [86](#)

verify_and_link

- seq64::event_list, [34](#)
- seq64::sequence, [165](#)

WHITE

- seq64::font, [36](#)

write

- seq64::midifile, [83](#)
- seq64::optionsfile, [90](#)
- seq64::userfile, [182](#)

write_byte

- seq64::midifile, [85](#)

write_long

- seq64::midifile, [84](#)

write_prop_header

- seq64::midifile, [85](#)

write_proprietary_track

- seq64::midifile, [86](#)

write_seq_number

- seq64::midifile, [85](#)

write_short

- seq64::midifile, [85](#)

write_track_name

- seq64::midifile, [85](#)

write_varinum

- seq64::midifile, [85](#)

x_to_w

- seq64::seqevent, [137](#)

xy_to_rect

- seq64::seqdata, [126](#)

YELLOW_ON_BLACK

- seq64::font, [36](#)

zero_markers

- seq64::sequence, [165](#)