

Sequencer64 Developer/Tester's Reference Manual

0.9.9.10

Generated by Doxygen 1.8.9.1

Mon Nov 23 2015 20:40:32

Contents

1	Sequencer64	1
1.1	Introduction	1
2	MIDI File Parsing in Sequencer64	3
2.1	Introduction	3
2.2	SMF 1 Parsing	3
2.2.1	MIDI File Header, MThd	3
2.2.2	MIDI Track, MTrk	4
2.2.2.1	Channel Events	4
2.2.2.2	Meta Events	5
2.2.2.3	SysEx Events	5
2.2.2.4	End of Sequence	5
2.3	SMF 0 Parsing	5
3	User Testing of Sequencer64 with Yoshimi	7
3.1	Introduction	7
3.2	Smoke Test	7
3.3	Tests in the Patterns Window	8
3.3.1	Button Clicks on a Pattern	8
3.3.2	Patterns Window Key Shortcuts	9
3.3.3	The Sequencer64 User File	9
3.4	Tests Using Valgrind	9
3.4.1	Valgrind Suppressions	9
3.4.2	Full Valgrind Leak-Checking	10
3.4.2.1	Leak-Checking Basic Operation	10
3.5	Specific Fault Debugging	10
3.6	Snipping of a MIDI file.	10
4	Licenses	13
4.1	License Terms for the This Project.	13
4.2	XPC Application License	13
4.3	XPC Library License	14

4.4	XPC Documentation License	14
4.5	XPC Affero License	14
4.6	XPC License Summary	15
5	Todo List	17
6	Deprecated List	19
7	Hierarchical Index	21
7.1	Class Hierarchy	21
8	Data Structure Index	23
8.1	Data Structures	23
9	Data Structure Documentation	27
9.1	seq64::AbstractPerfInput Class Reference	27
9.2	seq64::click Class Reference	27
9.2.1	Detailed Description	28
9.2.2	Constructor & Destructor Documentation	29
9.2.2.1	click	29
9.2.2.2	click	29
9.2.2.3	click	29
9.2.3	Member Function Documentation	29
9.2.3.1	operator=	29
9.2.4	Field Documentation	29
9.2.4.1	m_x	29
9.2.4.2	m_y	29
9.2.4.3	m_button	29
9.2.4.4	m_modifier	30
9.3	seq64::configfile Class Reference	30
9.3.1	Constructor & Destructor Documentation	31
9.3.1.1	configfile	31
9.3.2	Member Function Documentation	31
9.3.2.1	next_data_line	31
9.3.2.2	line_after	31
9.3.3	Field Documentation	31
9.3.3.1	m_line	31
9.4	seq64::event Class Reference	31
9.4.1	Detailed Description	34
9.4.2	Constructor & Destructor Documentation	34
9.4.2.1	~event	34
9.4.3	Member Function Documentation	34

9.4.3.1	operator<	34
9.4.3.2	is_channel_msg	35
9.4.3.3	is_one_byte_msg	35
9.4.3.4	is_two_byte_msg	35
9.4.3.5	is_desired_cc_or_not_cc	35
9.4.3.6	mod_timestamp	36
9.4.3.7	set_status	36
9.4.3.8	set_data	36
9.4.3.9	set_data	36
9.4.3.10	get_data	37
9.4.3.11	append_sysex	37
9.4.3.12	get_rank	37
9.4.4	Field Documentation	37
9.4.4.1	m_status	37
9.4.4.2	m_data	37
9.4.4.3	m_sysex	37
9.4.4.4	m_has_link	38
9.5	seq64::event_list::event_key Class Reference	38
9.5.1	Detailed Description	38
9.5.2	Constructor & Destructor Documentation	38
9.5.2.1	event_key	38
9.5.2.2	event_key	38
9.5.3	Member Function Documentation	38
9.5.3.1	operator<	38
9.5.4	Field Documentation	39
9.5.4.1	m_timestamp	39
9.5.4.2	m_rank	39
9.6	seq64::event_list Class Reference	39
9.6.1	Detailed Description	41
9.6.2	Constructor & Destructor Documentation	41
9.6.2.1	event_list	41
9.6.3	Member Function Documentation	41
9.6.3.1	operator=	41
9.6.3.2	count	41
9.6.3.3	add	41
9.6.3.4	merge	41
9.6.3.5	link_new	42
9.6.3.6	verify_and_link	42
9.6.3.7	mark_out_of_range	42
9.6.3.8	any_selected_notes	42

9.6.3.9	count_selected_events	42
9.7	seq64::font Class Reference	42
9.7.1	Member Enumeration Documentation	44
9.7.1.1	Color	44
9.7.2	Member Function Documentation	44
9.7.2.1	init	44
9.7.2.2	render_string_on_drawable	44
9.7.3	Field Documentation	44
9.7.3.1	m_font_w	44
9.7.3.2	m_font_h	44
9.7.3.3	m_pixmap	45
9.7.3.4	m_black_pixmap	45
9.7.3.5	m_white_pixmap	45
9.7.3.6	m_b_on_y_pixmap	45
9.7.3.7	m_y_on_b_pixmap	45
9.7.3.8	m_clip_mask	45
9.8	seq64::gui_assistant Class Reference	45
9.8.1	Detailed Description	46
9.8.2	Constructor & Destructor Documentation	47
9.8.2.1	gui_assistant	47
9.9	seq64::gui_assistant_gtk2 Class Reference	47
9.9.1	Member Function Documentation	48
9.9.1.1	lash_timeout_connect	48
9.9.2	Field Documentation	48
9.9.2.1	sm_internal_keys	48
9.10	seq64::gui_drawingarea_gtk2 Class Reference	48
9.10.1	Detailed Description	51
9.10.2	Member Function Documentation	51
9.10.2.1	clear_window	51
9.10.2.2	set_line	51
9.10.2.3	draw_line	51
9.10.2.4	draw_line	51
9.10.2.5	draw_line_on_pixmap	51
9.10.2.6	draw_line_on_pixmap	52
9.10.2.7	draw_line	52
9.10.2.8	draw_line	52
9.10.2.9	draw_line	52
9.10.2.10	draw_line	53
9.10.2.11	render_string	53
9.10.2.12	render_string_on_pixmap	53

9.10.2.13 draw_rectangle	53
9.10.2.14 draw_rectangle	53
9.10.2.15 draw_rectangle	54
9.10.2.16 draw_rectangle	54
9.10.2.17 draw_rectangle	54
9.10.2.18 draw_rectangle	55
9.10.2.19 draw_rectangle_on_pixmap	55
9.10.2.20 draw_rectangle_on_pixmap	55
9.10.2.21 draw_normal_rectangle_on_pixmap	55
9.10.2.22 on_realize	56
9.10.3 Field Documentation	56
9.10.3.1 m_window	56
9.10.3.2 m_pixmap	56
9.10.3.3 m_background	56
9.10.3.4 m_foreground	56
9.10.3.5 m_mainperf	56
9.10.3.6 m_window_x	56
9.10.3.7 m_drop_x	56
9.11 seq64::gui_palette_gtk2 Class Reference	56
9.11.1 Detailed Description	57
9.11.2 Constructor & Destructor Documentation	57
9.11.2.1 gui_palette_gtk2	57
9.12 seq64::gui_window_gtk2 Class Reference	58
9.12.1 Constructor & Destructor Documentation	59
9.12.1.1 gui_window_gtk2	59
9.12.2 Field Documentation	59
9.12.2.1 m_window_x	59
9.13 seq64::jack_assistant Class Reference	59
9.13.1 Constructor & Destructor Documentation	60
9.13.1.1 jack_assistant	60
9.13.2 Member Function Documentation	60
9.13.2.1 init	60
9.13.2.2 stop	60
9.13.2.3 position	60
9.13.2.4 output	61
9.13.2.5 info_message	61
9.13.2.6 error_message	61
9.13.3 Friends And Related Function Documentation	61
9.13.3.1 jack_sync_callback	61
9.13.3.2 jack_shutdown	61

9.13.3.3	jack_timebase_callback	61
9.14	seq64::jack_scratchpad Struct Reference	62
9.14.1	Detailed Description	62
9.15	seq64::keybindentry Class Reference	62
9.15.1	Member Enumeration Documentation	63
9.15.1.1	type	63
9.15.2	Constructor & Destructor Documentation	63
9.15.2.1	keybindentry	63
9.15.3	Member Function Documentation	63
9.15.3.1	set	63
9.15.3.2	on_key_press_event	63
9.15.4	Field Documentation	63
9.15.4.1	m_key	63
9.16	seq64::keys_perform Class Reference	63
9.16.1	Detailed Description	65
9.16.2	Constructor & Destructor Documentation	65
9.16.2.1	~keys_perform	65
9.16.3	Member Function Documentation	65
9.16.3.1	set_keys	65
9.16.3.2	get_keys	66
9.16.3.3	show_ui_sequence_key	66
9.16.3.4	show_ui_sequence_number	66
9.16.3.5	key_name	66
9.16.3.6	set_all_key_events	66
9.16.3.7	set_all_key_groups	66
9.16.3.8	set_key_event	66
9.16.3.9	set_key_group	67
9.16.4	Field Documentation	67
9.16.4.1	m_key_show_ui_sequence_number	67
9.16.4.2	m_key_bpm_up	67
9.17	seq64::keys_perform_gtk2 Class Reference	67
9.17.1	Detailed Description	69
9.17.2	Constructor & Destructor Documentation	69
9.17.2.1	~keys_perform_gtk2	69
9.17.3	Member Function Documentation	69
9.17.3.1	key_name	69
9.17.3.2	set_all_key_events	69
9.17.3.3	set_all_key_groups	69
9.18	seq64::keys_perform_transfer Struct Reference	69
9.19	seq64::keystroke Class Reference	69

9.19.1 Detailed Description	70
9.19.2 Constructor & Destructor Documentation	70
9.19.2.1 keystroke	70
9.19.2.2 keystroke	71
9.19.3 Member Function Documentation	71
9.19.3.1 operator=	71
9.19.3.2 is_letter	71
9.19.4 Field Documentation	71
9.19.4.1 m_is_press	71
9.19.4.2 m_key	71
9.19.4.3 m_modifier	72
9.20 seq64::lash Class Reference	72
9.20.1 Detailed Description	72
9.20.2 Constructor & Destructor Documentation	72
9.20.2.1 lash	72
9.20.3 Member Function Documentation	73
9.20.3.1 set_alsa_client_id	73
9.20.3.2 process_events	73
9.20.3.3 init	73
9.20.3.4 handle_event	73
9.20.3.5 handle_config	73
9.21 seq64::maintime Class Reference	73
9.21.1 Detailed Description	75
9.21.2 Constructor & Destructor Documentation	75
9.21.2.1 maintime	75
9.21.3 Member Function Documentation	76
9.21.3.1 idle_progress	76
9.21.3.2 on_realize	77
9.21.3.3 on_expose_event	77
9.21.4 Field Documentation	77
9.21.4.1 m_beat_width	77
9.21.4.2 m_bar_width	77
9.21.4.3 m_box_width	77
9.21.4.4 m_box_height	77
9.21.4.5 m_flash_width	77
9.21.4.6 m_flash_height	77
9.21.4.7 m_ppqn	77
9.22 seq64::mainwid Class Reference	78
9.22.1 Constructor & Destructor Documentation	80
9.22.1.1 mainwid	80

9.22.2	Member Function Documentation	80
9.22.2.1	set_screenset	80
9.22.2.2	update_sequence_on_window	80
9.22.2.3	update_markers	80
9.22.2.4	draw_marker_on_sequence	80
9.22.2.5	valid_sequence	81
9.22.2.6	draw_sequence_on_pixmap	81
9.22.2.7	draw_sequences_on_pixmap	81
9.22.2.8	draw_sequence_pixmap_on_window	81
9.22.2.9	seq_from_xy	81
9.22.2.10	timeout	82
9.22.2.11	redraw	82
9.22.2.12	calculate_base_sizes	82
9.22.2.13	on_realize	82
9.22.2.14	on_expose_event	82
9.22.2.15	on_button_press_event	83
9.22.2.16	on_button_release_event	83
9.22.2.17	on_motion_notify_event	83
9.22.2.18	on_focus_in_event	83
9.22.2.19	on_focus_out_event	84
9.22.3	Field Documentation	84
9.22.3.1	m_screenset_slots	84
9.22.3.2	m_screenset_offset	84
9.23	seq64::mainwnd Class Reference	84
9.23.1	Constructor & Destructor Documentation	88
9.23.1.1	mainwnd	88
9.23.2	Member Function Documentation	88
9.23.2.1	open_file	88
9.23.2.2	ppqn	88
9.23.2.3	file_import_dialog	89
9.23.2.4	about_dialog	89
9.23.2.5	adj_callback_ss	89
9.23.2.6	adj_callback_bpm	89
9.23.2.7	edit_callback_notepad	89
9.23.2.8	timer_callback	89
9.23.2.9	open_performance_edit	89
9.23.2.10	update_window_title	90
9.23.2.11	new_file	90
9.23.2.12	save_file	90
9.23.2.13	signal_action	90

9.23.2.14	on_delete_event	90
9.23.2.15	on_key_press_event	90
9.23.2.16	on_key_release_event	90
9.23.2.17	on_grouplearnchange	90
9.23.3	Field Documentation	91
9.23.3.1	m_sigpipe	91
9.23.3.2	m_main_wid	91
9.23.3.3	m_spinbutton_load_offset	91
9.24	seq64::midi_container Class Reference	91
9.24.1	Member Function Documentation	92
9.24.1.1	fill	92
9.24.1.2	put	93
9.24.1.3	get	93
9.24.1.4	position	93
9.24.1.5	add_variable	93
9.24.1.6	add_long	93
9.25	seq64::midi_list Class Reference	93
9.25.1	Member Typedef Documentation	95
9.25.1.1	CharList	95
9.25.2	Member Function Documentation	95
9.25.2.1	put	95
9.25.2.2	get	95
9.26	seq64::midi_vector Class Reference	95
9.26.1	Member Function Documentation	97
9.26.1.1	put	97
9.26.1.2	get	97
9.27	seq64::midibus Class Reference	97
9.27.1	Member Function Documentation	99
9.27.1.1	init_out	99
9.27.1.2	init_in	99
9.27.1.3	deinit_in	99
9.27.1.4	init_out_sub	99
9.27.1.5	init_in_sub	100
9.27.1.6	play	100
9.27.1.7	sysex	100
9.27.1.8	clock	100
9.27.1.9	continue_from	100
9.27.1.10	init_clock	100
9.27.1.11	set_input	100
9.28	seq64::midifile Class Reference	100

9.28.1	Detailed Description	103
9.28.2	Constructor & Destructor Documentation	103
9.28.2.1	midfile	103
9.28.3	Member Function Documentation	104
9.28.3.1	parse	104
9.28.3.2	write	105
9.28.3.3	ppqn	106
9.28.3.4	smf0_increment	106
9.28.3.5	parse_smf_0	106
9.28.3.6	parse_smf_1	106
9.28.3.7	parse_prop_header	106
9.28.3.8	parse_proprietary_track	107
9.28.3.9	read_long	108
9.28.3.10	read_varinum	108
9.28.3.11	write_long	108
9.28.3.12	write_short	108
9.28.3.13	read_byte_array	108
9.28.3.14	write_byte	108
9.28.3.15	write_varinum	108
9.28.3.16	write_track_name	109
9.28.3.17	read_track_name	109
9.28.3.18	write_seq_number	109
9.28.3.19	read_seq_number	109
9.28.3.20	write_prop_header	109
9.28.3.21	write_proprietary_track	110
9.28.3.22	varinum_size	110
9.28.3.23	prop_item_size	111
9.28.3.24	errdump	111
9.28.3.25	errdump	111
9.28.3.26	is_sysex_special_id	111
9.28.4	Field Documentation	112
9.28.4.1	m_file_size	112
9.28.4.2	m_error_message	112
9.28.4.3	m_disable_reported	112
9.28.4.4	m_pos	112
9.28.4.5	m_data	112
9.28.4.6	m_char_list	112
9.28.4.7	m_new_format	112
9.28.4.8	m_smf0_channels_count	112
9.28.4.9	m_smf0_channels	112

9.29	seq64::options Class Reference	112
9.29.1	Field Documentation	113
9.29.1.1	m_notebook	113
9.30	seq64::optionsfile Class Reference	113
9.30.1	Detailed Description	114
9.30.2	Member Function Documentation	114
9.30.2.1	parse	114
9.30.2.2	write	116
9.31	seq64::perfedit Class Reference	116
9.31.1	Detailed Description	119
9.31.2	Constructor & Destructor Documentation	119
9.31.2.1	perfedit	119
9.31.2.2	~perfedit	119
9.31.3	Member Function Documentation	119
9.31.3.1	init_before_show	119
9.31.3.2	set_beats_per_bar	119
9.31.3.3	set_beat_width	119
9.31.3.4	set_guides	119
9.31.3.5	grow	120
9.31.3.6	expand	120
9.31.3.7	collapse	120
9.31.3.8	copy	120
9.31.3.9	undo	120
9.31.3.10	timeout	120
9.31.3.11	start_playing	120
9.31.3.12	stop_playing	120
9.31.4	Field Documentation	120
9.31.4.1	m_bpm	120
9.31.4.2	m_bw	120
9.31.4.3	m_redraw_ms	120
9.32	seq64::perfnames Class Reference	121
9.32.1	Detailed Description	123
9.32.2	Constructor & Destructor Documentation	123
9.32.2.1	perfnames	123
9.32.3	Member Function Documentation	123
9.32.3.1	draw_sequence	123
9.32.3.2	on_realize	123
9.32.3.3	on_expose_event	123
9.32.3.4	on_size_allocate	123
9.32.4	Field Documentation	123

9.32.4.1	m_names_chars	123
9.32.4.2	m_char_w	123
9.32.4.3	m_setbox_w	123
9.32.4.4	m_namebox_w	123
9.32.4.5	m_names_y	124
9.32.4.6	m_xy_offset	124
9.33	seq64::perform Class Reference	124
9.33.1	Detailed Description	130
9.33.2	Constructor & Destructor Documentation	130
9.33.2.1	perform	130
9.33.2.2	~perform	131
9.33.3	Member Function Documentation	131
9.33.3.1	modify	131
9.33.3.2	sequence_count	131
9.33.3.3	init	131
9.33.3.4	clear_all	131
9.33.3.5	launch_input_thread	131
9.33.3.6	launch_output_thread	131
9.33.3.7	init_jack	131
9.33.3.8	new_sequence	131
9.33.3.9	add_sequence	132
9.33.3.10	delete_sequence	132
9.33.3.11	clear_sequence_triggers	132
9.33.3.12	set_left_tick	132
9.33.3.13	set_right_tick	133
9.33.3.14	move_triggers	133
9.33.3.15	copy_triggers	133
9.33.3.16	push_trigger_undo	133
9.33.3.17	get_max_trigger	133
9.33.3.18	midi_control_toggle	133
9.33.3.19	midi_control_on	133
9.33.3.20	midi_control_off	134
9.33.3.21	handle_midi_control	134
9.33.3.22	get_screen_set_notepad	134
9.33.3.23	set_screen_set_notepad	134
9.33.3.24	set_screenset	134
9.33.3.25	set_playing_screenset	135
9.33.3.26	select_group_mute	135
9.33.3.27	unset_mode_group_learn	135
9.33.3.28	select_mute_group	135

9.33.3.29 start	135
9.33.3.30 stop	135
9.33.3.31 all_notes_off	135
9.33.3.32 set_active	135
9.33.3.33 set_was_active	136
9.33.3.34 is_dirty_main	136
9.33.3.35 is_dirty_edit	136
9.33.3.36 is_dirty_perf	136
9.33.3.37 is_dirty_names	136
9.33.3.38 is_active	137
9.33.3.39 get_sequence	137
9.33.3.40 reset_sequences	137
9.33.3.41 play	137
9.33.3.42 set_orig_ticks	137
9.33.3.43 set_beats_per_minute	138
9.33.3.44 set_sequence_control_status	138
9.33.3.45 unset_sequence_control_status	138
9.33.3.46 sequence_playing_on	138
9.33.3.47 sequence_playing_off	138
9.33.3.48 set_group_mute_state	138
9.33.3.49 get_group_mute_state	138
9.33.3.50 output_func	139
9.33.3.51 set_offset	139
9.33.3.52 save_playing_state	139
9.33.3.53 show_ui_sequence_key	139
9.33.3.54 show_ui_sequence_number	139
9.33.3.55 start_playing	139
9.33.3.56 decrement_beats_per_minute	139
9.33.3.57 increment_beats_per_minute	139
9.33.3.58 highlight	139
9.33.3.59 sequence_label	140
9.33.3.60 set_input_bus	140
9.33.3.61 mainwnd_key_event	140
9.33.3.62 perfroll_key_event	140
9.33.3.63 seq_in_playing_screen	141
9.33.3.64 is_modified	142
9.33.3.65 is_midi_control_valid	142
9.33.3.66 is_screenset_valid	142
9.33.3.67 is_seq_valid	142
9.33.3.68 is_mseq_valid	142

9.33.3.69	install_sequence	143
9.33.3.70	inner_start	143
9.33.3.71	clamp_track	143
9.33.3.72	set_key_event	143
9.33.3.73	set_key_group	143
9.33.4	Friends And Related Function Documentation	143
9.33.4.1	jack_sync_callback	143
9.33.5	Field Documentation	144
9.33.5.1	sm_mc_dummy	144
9.33.5.2	m_playscreen_offset	144
9.33.5.3	m_seqs	144
9.33.5.4	m_playback_mode	144
9.33.5.5	m_one_measure	144
9.33.5.6	m_left_tick	144
9.33.5.7	m_right_tick	144
9.33.5.8	m_starting_tick	144
9.33.5.9	m_seqs_in_set	145
9.33.5.10	m_max_sets	145
9.33.5.11	m_sequence_count	145
9.33.5.12	m_sequence_max	145
9.33.5.13	m_is_modified	145
9.34	seq64::performcallback Struct Reference	145
9.34.1	Detailed Description	146
9.35	seq64::perroll Class Reference	146
9.35.1	Constructor & Destructor Documentation	149
9.35.1.1	~perroll	149
9.35.2	Member Function Documentation	149
9.35.2.1	set_guides	149
9.35.2.2	update_sizes	150
9.35.2.3	init_before_show	150
9.35.2.4	fill_background_pixmap	150
9.35.2.5	set_ppqn	150
9.35.2.6	convert_xy	150
9.35.2.7	convert_x	150
9.35.2.8	snap_x	150
9.35.2.9	start_playing	151
9.35.2.10	stop_playing	151
9.35.2.11	draw_sequence_on	151
9.35.2.12	draw_drawable_row	151
9.35.2.13	on_realize	151

9.35.2.14	on_expose_event	151
9.35.2.15	on_button_press_event	151
9.35.2.16	on_button_release_event	151
9.35.2.17	on_key_press_event	151
9.35.3	Field Documentation	152
9.35.3.1	m_fruity_interaction	152
9.36	seq64::perftime Class Reference	152
9.36.1	Constructor & Destructor Documentation	155
9.36.1.1	perftime	155
9.36.2	Member Function Documentation	155
9.36.2.1	set_guides	155
9.36.2.2	on_realize	155
9.36.2.3	on_expose_event	155
9.36.3	Field Documentation	155
9.36.3.1	m_measure_length	155
9.36.3.2	m_left_marker_tick	156
9.36.3.3	m_right_marker_tick	156
9.37	seq64::rc_settings Class Reference	156
9.37.1	Member Function Documentation	158
9.37.1.1	home_config_directory	158
9.38	seq64::gui_drawingarea_gtk2::rect Struct Reference	158
9.39	seq64::rect Class Reference	158
9.40	seq64::Seq24PerfInput Class Reference	158
9.40.1	Member Function Documentation	160
9.40.1.1	on_button_press_event	160
9.40.1.2	on_button_release_event	160
9.40.1.3	on_motion_notify_event	160
9.40.1.4	set_adding	160
9.40.1.5	handle_motion_key	160
9.41	seq64::Seq24SeqEventInput Struct Reference	161
9.41.1	Member Function Documentation	161
9.41.1.1	set_adding	161
9.41.1.2	on_button_press_event	161
9.41.1.3	on_button_release_event	161
9.41.1.4	on_motion_notify_event	162
9.42	seq64::Seq24SeqRollInput Class Reference	162
9.42.1	Member Function Documentation	162
9.42.1.1	set_adding	162
9.42.1.2	on_button_press_event	162
9.42.1.3	on_button_release_event	162

9.43 seq64::seqdata Class Reference	162
9.43.1 Constructor & Destructor Documentation	165
9.43.1.1 seqdata	165
9.43.2 Member Function Documentation	165
9.43.2.1 reset	165
9.43.2.2 redraw	165
9.43.2.3 set_zoom	165
9.43.2.4 set_data_type	165
9.43.2.5 idle_redraw	165
9.43.2.6 update_sizes	165
9.43.2.7 xy_to_rect	166
9.43.2.8 on_realize	166
9.43.2.9 on_button_press_event	166
9.43.2.10 on_button_release_event	166
9.43.2.11 on_motion_notify_event	166
9.43.2.12 on_scroll_event	166
9.43.3 Field Documentation	166
9.43.3.1 m_number_w	166
9.43.3.2 m_number_h	166
9.43.3.3 m_number_offset_y	167
9.44 seq64::seqedit Class Reference	167
9.44.1 Detailed Description	170
9.44.2 Constructor & Destructor Documentation	170
9.44.2.1 seqedit	170
9.44.3 Member Function Documentation	170
9.44.3.1 set_zoom	170
9.44.3.2 set_snap	170
9.44.3.3 set_note_length	171
9.44.3.4 set_measures	171
9.44.3.5 apply_length	171
9.44.3.6 get_measures	171
9.44.3.7 set_midi_channel	171
9.44.3.8 set_midi_bus	171
9.44.3.9 set_scale	171
9.44.3.10 set_key	172
9.44.3.11 set_background_sequence	172
9.44.3.12 name_change_callback	172
9.44.3.13 set_data_type	172
9.44.3.14 create_menus	172
9.44.3.15 popup_event_menu	173

9.44.3.16	popup_midibus_menu	173
9.44.3.17	popup_sequence_menu	173
9.44.3.18	popup_tool_menu	173
9.44.3.19	do_action	173
9.44.3.20	on_delete_event	173
9.44.4	Field Documentation	173
9.44.4.1	m_initial_snap	173
9.45	seq64::seqevent Class Reference	174
9.45.1	Member Function Documentation	177
9.45.1.1	set_snap	177
9.45.1.2	set_data_type	177
9.45.1.3	update_sizes	177
9.45.1.4	draw_background	177
9.45.1.5	draw_pixmap_on_window	177
9.45.1.6	idle_redraw	178
9.45.1.7	x_to_w	178
9.45.1.8	drop_event	178
9.45.1.9	draw_events_on	178
9.45.1.10	start_paste	178
9.45.1.11	convert_x	178
9.45.1.12	convert_t	178
9.45.1.13	snap_x	178
9.45.1.14	on_realize	178
9.45.1.15	on_button_press_event	178
9.45.1.16	on_button_release_event	179
9.45.1.17	on_motion_notify_event	179
9.45.1.18	on_key_press_event	179
9.46	seq64::seqkeys Class Reference	179
9.46.1	Member Function Documentation	182
9.46.1.1	set_hint_state	182
9.46.1.2	draw_key	183
9.46.1.3	on_realize	183
9.46.1.4	on_button_press_event	183
9.46.1.5	on_button_release_event	183
9.46.1.6	on_motion_notify_event	183
9.46.1.7	on_enter_notify_event	183
9.46.1.8	on_leave_notify_event	183
9.47	seq64::seqmenu Class Reference	184
9.47.1	Detailed Description	185
9.47.2	Constructor & Destructor Documentation	186

9.47.2.1	seqmenu	186
9.47.2.2	~seqmenu	186
9.47.3	Member Function Documentation	186
9.47.3.1	popup_menu	186
9.47.3.2	seq_edit	186
9.47.3.3	seq_copy	186
9.47.3.4	seq_cut	186
9.47.3.5	seq_paste	186
9.47.3.6	seq_clear_perf	186
9.47.4	Field Documentation	187
9.47.4.1	m_seqedit	187
9.48	seq64::seqroll Class Reference	187
9.48.1	Member Function Documentation	191
9.48.1.1	reset	191
9.48.1.2	redraw_events	191
9.48.1.3	set_data_type	191
9.48.1.4	set_background_sequence	191
9.48.1.5	update_sizes	191
9.48.1.6	update_background	191
9.48.1.7	draw_events_on_pixmap	192
9.48.1.8	convert_tn	192
9.48.1.9	snap_x	192
9.48.1.10	draw_events_on	192
9.48.1.11	on_key_press_event	192
9.48.1.12	on_scroll_event	192
9.49	seq64::seqtime Class Reference	192
9.49.1	Member Function Documentation	194
9.49.1.1	on_button_press_event	194
9.49.1.2	on_button_release_event	194
9.50	seq64::sequence Class Reference	194
9.50.1	Detailed Description	201
9.50.2	Member Enumeration Documentation	201
9.50.2.1	select_action_e	201
9.50.3	Member Function Documentation	202
9.50.3.1	operator=	202
9.50.3.2	event_count	202
9.50.3.3	push_undo	202
9.50.3.4	pop_undo	202
9.50.3.5	pop_redo	202
9.50.3.6	push_trigger_undo	202

9.50.3.7	set_beats_per_bar	202
9.50.3.8	set_beat_width	202
9.50.3.9	get_beat_width	202
9.50.3.10	set_rec_vol	202
9.50.3.11	get_name	203
9.50.3.12	set_length	203
9.50.3.13	set_playing	203
9.50.3.14	toggle_queued	203
9.50.3.15	off_queued	203
9.50.3.16	set_recording	203
9.50.3.17	set_snap_tick	203
9.50.3.18	set_quantized_rec	204
9.50.3.19	set_thru	204
9.50.3.20	is_dirty_main	204
9.50.3.21	is_dirty_edit	204
9.50.3.22	is_dirty_perf	204
9.50.3.23	is_dirty_names	204
9.50.3.24	set_dirty_mp	204
9.50.3.25	set_dirty	204
9.50.3.26	set_midi_channel	204
9.50.3.27	print	204
9.50.3.28	print_triggers	204
9.50.3.29	play	204
9.50.3.30	set_orig_tick	205
9.50.3.31	add_event	205
9.50.3.32	add_trigger	205
9.50.3.33	split_trigger	205
9.50.3.34	grow_trigger	205
9.50.3.35	del_trigger	205
9.50.3.36	get_trigger_state	206
9.50.3.37	select_trigger	207
9.50.3.38	unselect_triggers	207
9.50.3.39	intersect_triggers	207
9.50.3.40	intersect_notes	207
9.50.3.41	intersect_events	208
9.50.3.42	paste_trigger	208
9.50.3.43	move_selected_triggers_to	208
9.50.3.44	selected_trigger_start	208
9.50.3.45	selected_trigger_end	209
9.50.3.46	get_max_trigger	209

9.50.3.47 move_triggers	209
9.50.3.48 copy_triggers	209
9.50.3.49 clear_triggers	209
9.50.3.50 set_midi_bus	209
9.50.3.51 set_master_midi_bus	209
9.50.3.52 select_note_events	209
9.50.3.53 select_events	209
9.50.3.54 select_events	209
9.50.3.55 get_num_selected_notes	210
9.50.3.56 get_num_selected_events	210
9.50.3.57 select_all	210
9.50.3.58 copy_selected	210
9.50.3.59 paste_selected	210
9.50.3.60 add_note	210
9.50.3.61 add_event	210
9.50.3.62 stream_event	210
9.50.3.63 change_event_data_range	210
9.50.3.64 increment_selected	211
9.50.3.65 decrement_selected	211
9.50.3.66 grow_selected	212
9.50.3.67 stretch_selected	212
9.50.3.68 remove_marked	212
9.50.3.69 mark_selected	212
9.50.3.70 unpaint_all	212
9.50.3.71 unselect	212
9.50.3.72 verify_and_link	212
9.50.3.73 link_new	212
9.50.3.74 zero_markers	212
9.50.3.75 play_note_on	212
9.50.3.76 play_note_off	212
9.50.3.77 off_playing_notes	213
9.50.3.78 reset_draw_marker	213
9.50.3.79 reset_draw_trigger_marker	213
9.50.3.80 get_next_note_event	213
9.50.3.81 get_lowest_note_event	213
9.50.3.82 get_highest_note_event	213
9.50.3.83 get_next_event	213
9.50.3.84 get_next_event	213
9.50.3.85 fill_container	213
9.50.3.86 transpose_notes	214

9.50.3.87 background_sequence	214
9.50.3.88 put_event_on_bus	214
9.50.3.89 set_trigger_offset	214
9.50.3.90 split_trigger	214
9.50.3.91 adjust_trigger_offsets_to_length	214
9.50.3.92 remove	214
9.50.3.93 remove	214
9.50.4 Field Documentation	215
9.50.4.1 m_playing_notes	215
9.50.4.2 m_seq_number	215
9.50.4.3 m_length	215
9.50.4.4 m_musical_key	215
9.50.4.5 m_musical_scale	215
9.50.4.6 m_background_sequence	215
9.50.4.7 m_mutex	215
9.51 seq64::trigger Class Reference	215
9.51.1 Detailed Description	216
9.52 seq64::triggers Class Reference	216
9.52.1 Constructor & Destructor Documentation	219
9.52.1.1 triggers	219
9.52.2 Member Function Documentation	220
9.52.2.1 operator=	220
9.52.2.2 set_length	220
9.52.2.3 play	220
9.52.2.4 add	220
9.52.2.5 adjust_offsets_to_length	221
9.52.2.6 split	221
9.52.2.7 split	221
9.52.2.8 grow	221
9.52.2.9 remove	221
9.52.2.10 get_state	222
9.52.2.11 select	222
9.52.2.12 unselect	222
9.52.2.13 intersect	222
9.52.2.14 paste	222
9.52.2.15 move_selected	223
9.52.2.16 get_selected_start	224
9.52.2.17 get_selected_end	224
9.52.2.18 copy	224
9.52.2.19 next	225

9.52.2.20	next_trigger	225
9.52.2.21	adjust_offset	225
9.52.3	Field Documentation	225
9.52.3.1	m_ppqn	225
9.52.3.2	m_length	225
9.53	seq64::user_instrument Class Reference	226
9.53.1	Detailed Description	226
9.53.2	Member Function Documentation	227
9.53.2.1	set_defaults	227
9.53.2.2	controller_max	227
9.53.2.3	controller_name	227
9.53.2.4	controller_active	227
9.53.2.5	set_controller	227
9.53.2.6	set_name	227
9.53.2.7	copy_definitions	227
9.53.3	Field Documentation	228
9.53.3.1	m_is_valid	228
9.53.3.2	m_controller_count	228
9.54	seq64::user_instrument_t Struct Reference	228
9.54.1	Field Documentation	228
9.54.1.1	instrument	228
9.54.1.2	controllers	228
9.54.1.3	controllers_active	228
9.55	seq64::user_midi_bus Class Reference	228
9.55.1	Detailed Description	229
9.55.2	Member Function Documentation	229
9.55.2.1	set_defaults	229
9.55.2.2	channel_count	230
9.55.2.3	channel_max	230
9.55.2.4	instrument	230
9.55.2.5	set_instrument	230
9.55.2.6	copy_definitions	230
9.55.3	Field Documentation	230
9.55.3.1	m_is_valid	230
9.55.3.2	m_channel_count	230
9.56	seq64::user_midi_bus_t Struct Reference	230
9.56.1	Field Documentation	231
9.56.1.1	alias	231
9.56.1.2	instrument	231
9.57	seq64::user_settings Class Reference	231

9.57.1	Detailed Description	237
9.57.2	Member Typedef Documentation	237
9.57.2.1	Busses	237
9.57.2.2	Instruments	237
9.57.3	Member Enumeration Documentation	237
9.57.3.1	mainwid_grid_style_t	237
9.57.4	Constructor & Destructor Documentation	238
9.57.4.1	user_settings	238
9.57.5	Member Function Documentation	238
9.57.5.1	set_defaults	238
9.57.5.2	set_globals	238
9.57.5.3	get_globals	238
9.57.5.4	bus	238
9.57.5.5	instrument	238
9.57.5.6	controller_active	238
9.57.5.7	controller_name	238
9.57.5.8	zoom	238
9.57.5.9	mainwnd_rows	238
9.57.5.10	mainwnd_cols	238
9.57.5.11	max_sets	239
9.57.5.12	text_x	239
9.57.5.13	text_y	239
9.57.5.14	mainwid_border	239
9.57.5.15	mainwid_spacing	239
9.57.5.16	control_height	239
9.57.5.17	dump_summary	239
9.57.5.18	midi_ppqn	239
9.57.5.19	midi_buss_override	239
9.57.5.20	midi_beats_per_bar	239
9.57.5.21	midi_beats_per_minute	239
9.57.5.22	midi_beat_width	240
9.57.5.23	private_bus	240
9.57.5.24	private_instrument	240
9.57.6	Field Documentation	240
9.57.6.1	m_midi_buses	240
9.57.6.2	m_instruments	240
9.57.6.3	m_grid_style	240
9.57.6.4	m_grid_brackets	240
9.57.6.5	m_mainwnd_rows	240
9.57.6.6	m_mainwnd_cols	240

9.57.6.7	<code>m_max_sets</code>	240
9.57.6.8	<code>m_mainwid_border</code>	241
9.57.6.9	<code>m_control_height</code>	241
9.57.6.10	<code>m_global_seq_feature_save</code>	241
9.57.6.11	<code>m_seqedit_scale</code>	241
9.57.6.12	<code>m_seqedit_key</code>	241
9.57.6.13	<code>m_seqedit_bgsequence</code>	241
9.57.6.14	<code>m_use_new_font</code>	241
9.57.6.15	<code>m_text_x</code>	241
9.57.6.16	<code>m_seqchars_x</code>	242
9.57.6.17	<code>m_midi_ppqn</code>	242
9.57.6.18	<code>m_midi_beats_per_measure</code>	242
9.57.6.19	<code>m_midi_beats_per_minute</code>	242
9.57.6.20	<code>m_midi_beat_width</code>	242
9.57.6.21	<code>m_midi_buss_override</code>	242
9.57.6.22	<code>m_seqs_in_set</code>	243
9.57.6.23	<code>m_gmute_tracks</code>	243
9.57.6.24	<code>m_max_sequence</code>	243
9.57.6.25	<code>m_seqarea_x</code>	243
9.57.6.26	<code>m_seqarea_seq_x</code>	243
9.57.6.27	<code>m_mainwid_x</code>	243
9.57.6.28	<code>m_save_user_config</code>	243
9.57.6.29	<code>mc_min_zoom</code>	243
9.57.6.30	<code>mc_max_zoom</code>	244
9.57.6.31	<code>mc_baseline_ppqn</code>	244
9.58	<code>seq64::userfile</code> Class Reference	244
9.58.1	Member Function Documentation	245
9.58.1.1	<code>parse</code>	245
9.58.1.2	<code>write</code>	245
9.58.1.3	<code>dump_setting_summary</code>	245
Index		247

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

MIDI File Parsing in Sequencer64

Author(s) Chris Ahlstrom 2015-11-23

2.1 Introduction

This section describes the parsing of a MIDI file. We start with the main format that is supported, SMF 1. Once we understand that completely, we will figure out how to split an SMF 0 track correctly.

We've already split the `midifile::parse()` function into two sections. The first section analyzes the header of the MIDI. Then, based on whether the file is SMF 1 (the normal case) or SMF 0, either the `parse_smf_1()` function or the `parse_smf_0()` function is called.

The `parse_smf_0()` function currently just calls `parse_smf_1()`. But our current thinking on `parse_smf_0()` is that it will create 17 sequence objects. The first one will contain the original track data, and the rest will contain common data and then channel data for each channel. After the parsing is done, all the non-empty tracks (including the original track) will be added to the performance. The user then has the option of deleting the original track, which will be the last track. That's the current idea, anyway.

2.2 SMF 1 Parsing

This section describes the parsing of the header chunk, MThd, and the track chunk, MTrk.

The `midifile::parse()` function starts by opening the MIDI file, getting its file-size, pre-allocating the data vector to that size, reading all of the characters into that vector, and then closing the file.

2.2.1 MIDI File Header, MThd

The data of the header is read:

Header ID:	"MThd"	<code>read_long()</code>	4 bytes
MThd length:	6	<code>read_long()</code>	4 bytes
Format:	0, 1, 2	<code>read_short()</code>	2 bytes
No. of track:	1 or more	<code>read_short()</code>	2 bytes
PPQN:	192	<code>read_short()</code>	2 bytes

The header ID and its length are always the same values. The formats that Sequencer64 supports are 0 or 1. SMF 0 has only one track, while SMF 1 can support an arbitrary number of tracks. The last value in the header is the PPQN value, which specifies the "pulses per quarter note", which is the basic time-resolution of events in the MIDI file. Common values are 96 or 192, but higher values are also common. Sequencer64 and its precursor, Seq24, default to 192.

2.2.2 MIDI Track, MTrk

Sequencer64 next reads the tracks specified in the file. Each track is assumed to cover a different MIDI channel, but always the same MIDI buss. (The MIDI buss is not a data item in standard MIDI files, but it is a special data item in Seq24/Sequencer64 MIDI files.) Each track is tagged by a standard chunk marker, "MTrk". Other markers are possible, and are to be ignored, if nothing else. Here are the values read at the beginning of a track:

Track ID:	"MTrk"	read_long()	4 bytes
Track length:	varies	read_long()	4 bytes

The track length is the number of bytes that need to be read in order to get all of the data in the track.

Next, a new sequence object is created, with the PPQN value passed to its constructor. The sequence then is hooked to the master MIDI buss object. The "RunningTime" accumulator is set to 0 for that track.

Next, the parse() function loops through the rest of the track, reading data and logging it to the sequence. Let's go through the loop, which is the meat of the processing.

TODO: An empty event is created before track processing, and re-used for every track and event. This seems dangerous. We moved the event constructor two levels of nesting deeper, and it seems to work fine.

Delta time. The amount time that passes from one event to the next is the *delta time*. For some events, the time doesn't matter, and is set to 0. This value is a *variable length value*, also known as a "VLV" or a "varinum". It provides a way of encoding arbitrarily large values, a byte at a time. For now, just note that a varinum is 1 or more bytes, and MIDI provides a way to tell when the varinum is complete.

Delta time:	varies	read_varinum()	1 or more bytes
-------------	--------	----------------	-----------------

2.2.2.1 Channel Events

Status. The byte after the delta time is examined by masking it against 0x80 to check the high bit. If not set, it is a "running status", it is replaced with the "last status", which is 0 at first.

Status byte:	varies	read_byte()	1 byte
--------------	--------	-------------	--------

If the high bit is set, it is a status, and is passed to the setter `event::set_status()`.

The "RunningTime" accumulator is incremented by the delta-time. The current time is adjusted as per the PPQN ratio, if needed, and passed to the setter `event::set_timestamp()`.

Now what does the status mean? First, the channel part of the status is masked out using the 0xF0 mask.

If it is a 2-data-byte event (note on, note off, aftertouch, control-change, or pitch-wheel), then the two data bytes are read:

Data byte 0:	varies	read_byte()	1 byte
Data byte 1:	varies	read_byte()	1 byte

If the status is a note-on event, with data[1] = 0, then it is converted to a note-off event, a fix for the output quirks of some MIDI devices, and the status of the event is amended to `EVENT_NOTE_OFF`.

If it is a 1-data-byte event (program change or channel pressure), then only data byte 0 is read.

Then the one or two data bytes are added to the event by overloads of `event::set_data()`, the event is added to the current sequence by `sequence::add_event()`, and the MIDI channel of the sequence is set by `sequence::set_midi_channel()`.

Note that this is the point where parsing could detect a change in channel, and select a new sequence to support that channel, and add the events to that sequence, if the file were SMF 0.

Also note that the channel of the sequence is set every a new channel event/status is read. This should be done once, and then simply warned about if a non-matching channel occurs.

Lastly, note that it might be better to do the sequence function calls at the end of processing the event.

2.2.2.2 Meta Events

If the event status masks off to 0xF0 (0xF0 to 0xFF), then it is a meta event. If the status is 0xFF, it is called a "Sequencer-specific", or "SeqSpec" event. For this kind of event, then a type byte and the length of the event are read.

Meta type:	varies	<code>read_byte()</code>	1 byte
Meta length:	varies	<code>read_varinum()</code>	1 or more bytes

If the type of the SeqSpec (0xFF) meta event is 0x7F, parsing checks to see if it is one of the Seq24 "proprietary" events. These events are tagged with various values that mask off to 0x24240000. The parser reads the tag:

Prop tag:	0x242400nn	<code>read_long()</code>	4 bytes
-----------	------------	--------------------------	---------

These tags provide a way to save and recover Seq24/Sequencer64 properties from the MIDI file: MIDI buss, MIDI channel, time signature, sequence triggers, and (new), the key, scale, and background sequence to use with the track/sequence. Any leftover data for the tagged event is let go. Unknown tags are skipped.

If the type of the SeqSpec (0xFF) meta event is 0x2f, then it is the End-of-Track marker. The current time is set using `sequence::set_length()` and then `sequence::zero_markers()` is called, and parsing is done for that track.

If the type of the SeqSpec (0xFF) meta event is 0x03, then it is the sequence name. The "length" number of bytes are read, and loaded by `sequence::set_name()`.

If the type of the SeqSpec (0xFF) meta event is 0x00, then it is the sequence number, which is read:

Seq number:	varies	<code>read_short()</code>	2 bytes
-------------	--------	---------------------------	---------

Note that the sequence number might be modified later to account for the current screenset in force for a file import operation.

Anything other SeqSpec type is simply skipped by reading the "length" number of bytes.

2.2.2.3 SysEx Events

If the meta event status value is 0xF0, it is called a "System-exclusive", or "SysEx" event.

Although there is some infrastructure to support storing the SysEx event within a sequence, the SysEx information is simply skipped. Sequencer64 warns if the terminating 0xF7 SysEx terminator is not found at the expected length. Also, some malformed SysEx messages have been encountered, and those are detected and skipped as well.

2.2.2.4 End of Sequence

Any other statuses are deemed unsupportable, and abort parsing with an error.

If the `-bus` option is in force, `sequence::set_midi_bus()` is called to override the buss number (if any) stored with the sequence.

Finally, `perform::add_sequence()` adds the sequence to the encoded tune.

2.3 SMF 0 Parsing

After parsing SMF 1 track data, we end up with a number of sequences, each on a different MIDI channel. With SMF 0, data for all channels is present in a single track. Sequencer64 will read SMF 0 data, but we really need to be able to have one MIDI channel per track. So we need to take the data from the sequence and use it to make more sequences.

TODO:

```
sequence::add_event()  sequence::set_midi_channel().  sequence::set_length()  
sequence::set_midi_bus() perform::add_sequence()
```


Chapter 3

User Testing of Sequencer64 with Yoshimi

Author(s) Chris Ahlstrom 2015-10-18

3.1 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.

3.2 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.

3.3 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.

3.3.1 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.

3.3.2 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.

3.3.3 The Sequencer64 User File

To be discussed.

3.4 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.

3.4.1 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>

3.4.2 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!)

3.4.2.1 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.

3.5 Specific Fault Debugging

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

3.6 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 4

Licenses

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

Author(s) Chris Ahlstrom 2015-09-10

4.1 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.

4.2 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>

4.3 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>

4.4 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>

4.5 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.

4.6 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 5

Todo List

File `calculations.cpp`

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.

File `perfnames.cpp`

When bringing up this dialog, and starting play from it, some extra horizontal lines are drawn for some of the sequences. This happens even in `seq24`, so this is long standing behavior. Is it useful, and how? Where is it done? In `perroll`?

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

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

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 ()`

Can we offload all this work to `perfedit`? Is it worthwhile?

Global `seq64::perfedit::perfedit (perform &p, int ppqn=SEQ64_USE_DEFAULT_PPQN)`

Offload most of the work into an initialization function like `options` does.

Global `seq64::perform::add_sequence (sequence *seq, int perf)`

Shouldn't we wrap around the sequence list if we can't find an empty sequence slot after `prefnum`?

Global `seq64::perform::is_active (int seq)`

We should have the sequence object keep track of its own activity and access that via a reference or pointer.

Global `seq64::perform::m_seqs [c_max_sequence]`

First, make the sequence array a vector, and second, put all of these flags into a structure and access those members indirectly.

Global `seq64::perform::set_beats_per_minute (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` (perform &perf, sequence &seq, int pos, int ppqn=SEQ64_USE_DEFAULT↔_PPQN)

Offload most of the work into an initialization function like options does.

Global `seq64::seqedit::set_background_sequence` (int 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 `seq64::triggers::next` (long *tick_on, long *tick_off, bool *selected, long *tick_offset)

It would be a bit simpler to simply return a trigger object, wouldn't it?

Chapter 6

Deprecated List

Global `seq64::sequence::get_name` () const

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

seq64::AbstractPerfInput	27
seq64::Seq24PerfInput	158
seq64::click	27
seq64::configfile	30
seq64::optionsfile	113
seq64::userfile	244
seq64::event	31
seq64::event_list::event_key	38
seq64::event_list	39
seq64::font	42
seq64::gui_assistant	45
seq64::gui_assistant_gtk2	47
seq64::gui_palette_gtk2	56
seq64::gui_drawingarea_gtk2	48
seq64::maintime	73
seq64::mainwid	78
seq64::perfnames	121
seq64::perfroll	146
seq64::perftime	152
seq64::seqdata	162
seq64::seqevent	174
seq64::seqkeys	179
seq64::seqroll	187
seq64::seqtime	192
seq64::gui_window_gtk2	58
seq64::mainwnd	84
seq64::perfedit	116
seq64::seqedit	167
seq64::jack_assistant	59
seq64::jack_scratchpad	62
seq64::keybindentry	62
seq64::keys_perform	63
seq64::keys_perform_gtk2	67
seq64::keys_perform_transfer	69
seq64::keystroke	69
seq64::lash	72

seq64::midi_container	91
seq64::midi_list	93
seq64::midi_vector	95
seq64::midibus	97
seq64::midifile	100
seq64::options	112
seq64::perform	124
seq64::performcallback	145
seq64::mainwnd	84
seq64::rc_settings	156
seq64::gui_drawingarea_gtk2::rect	158
seq64::rect	158
seq64::Seq24SeqEventInput	161
seq64::Seq24SeqRollInput	162
seq64::seqmenu	184
seq64::mainwid	78
seq64::perfnames	121
seq64::sequence	194
seq64::trigger	215
seq64::triggers	216
seq64::user_instrument	226
seq64::user_instrument_t	228
seq64::user_midi_bus	228
seq64::user_midi_bus_t	230
seq64::user_settings	231

Chapter 8

Data Structure Index

8.1 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 two "pills" that move in time with the beat and measure	??
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::midibus	Provides a class for handling the MIDI buss on Linux	??
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	??
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")	??
seq64::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 (Seq24) 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 receptable for a single track of MIDI data read from a MIDI file or edited into a pattern	??
seq64::trigger	This class hold a single trigger for a sequence object	??
seq64::triggers	Receptable the triggers that can be used with a sequence object	??
seq64::user_instrument	Provides data about the MIDI instruments, readable from the "user" configuration file	??
seq64::user_instrument_t	This structure corresponds to [user-instrument-N] definitions in the ~/.seq24usr or ~/.config/sequencer64/sequencer64.usr file	??
seq64::user_midi_bus	Provides data about the MIDI busses, readable from the "user" configuration file	??
seq64::user_midi_bus_t	This structure corresponds to [user-midi-bus-0] definitions in the ~/.seq24usr ("user") file (~/.config/sequencer64/sequencer64.usr in the latest version of the application)	??
seq64::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 ~/.config/sequencer64/sequencer64.usr and ~/.seq24usr configuration file	??

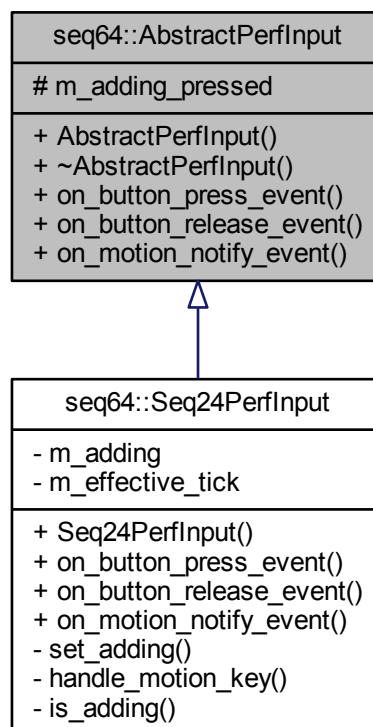
Chapter 9

Data Structure Documentation

9.1 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:



9.2 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.

9.2.1 Detailed Description

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

9.2.2 Constructor & Destructor Documentation

9.2.2.1 seq64::click::click ()

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

9.2.2.2 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.

9.2.2.3 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.
------------	--

9.2.3 Member Function Documentation

9.2.3.1 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.
------------	--

9.2.4 Field Documentation

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

0 is the left-most coordinate.

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

0 is the top-most coordinate.

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

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

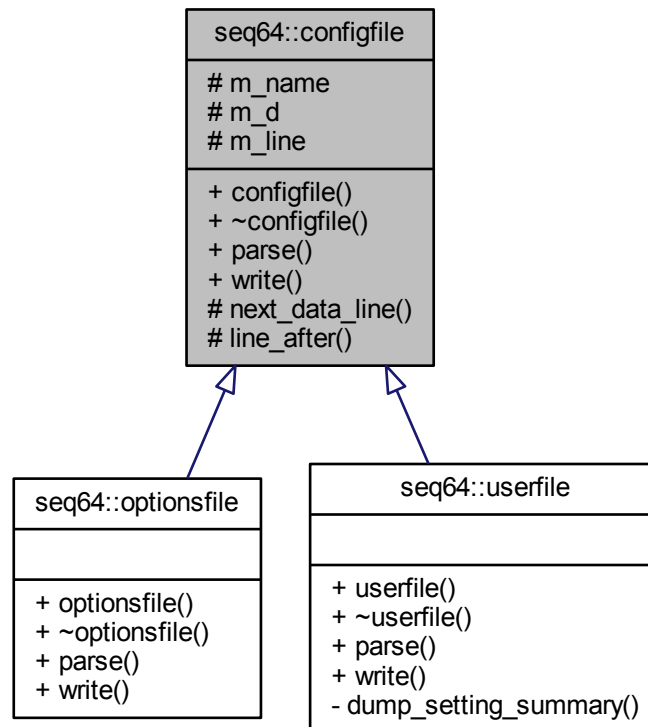
9.2.4.4 seq_modifier_t seq64::click::m_modifier [private]

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

9.3 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.
- `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.

9.3.1 Constructor & Destructor Documentation

9.3.1.1 `seq64::configfile::configfile (const std::string & name)`

Parameters

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

9.3.2 Member Function Documentation

9.3.2.1 `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.
---------------	----------------------------

9.3.2.2 `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.

9.3.3 Field Documentation

9.3.3.1 `char seq64::configfile::m_line[SEQ64_LINE_MAX]` `[protected]`

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

9.4 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`
- `midibyte 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 (midibyte status)`
Sets the `m_status` member to the value of `a_status`.
- `midibyte get_status () const`
'Getter' function for member `m_status`
- `void set_data (midibyte d1)`
Clears the most-significant-bit of the `d1` parameter, and sets it into the first byte of `m_data`.
- `void set_data (midibyte d1, midibyte d2)`
Clears the most-significant-bit of both parameters, and sets them into the first and second bytes of `m_data`.
- `void get_data (midibyte &d0, midibyte &d1) const`
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 (midibyte *data, long size)`
Appends SYSEX data to a new buffer.
- `midibyte * 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;.
- midibyte `data` (int index) const
'Getter' function for member m_data[]
- midibyte `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` (midibyte note)
Sets the note number, clearing off the most-significant-bit and assigning it to the first data byte, m_data[0].
- midibyte `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.

Static Public Member Functions

- static bool `is_channel_msg` (midibyte msg)
Static test for channel messages/statuses.
- static bool `is_one_byte_msg` (midibyte msg)
Static test for channel messages that have only one data byte.
- static bool `is_two_byte_msg` (midibyte msg)
Static test for channel messages that have two data bytes.
- static bool `is_desired_cc_or_not_cc` (midibyte msg, midibyte cc, midibyte datum)
Static test for channel messages that are either not control-change messages, or are and match the given controller value.

Private Attributes

- midibyte [m_status](#)
This is status byte without the channel.
- midibyte [m_data](#) [SEQ64_MIDI_DATA_BYTE_COUNT]
The two bytes of data for the MIDI event.
- midibyte * [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."

9.4.1 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.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 `seq64::event::~~event ()`

The [start_sysex\(\)](#) function does what we need.

9.4.3 Member Function Documentation

9.4.3.1 `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 `SEQ64_USE_EVENT_MAP` macro. (This actually works better than a list, for loading MIDI event, 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.

9.4.3.2 static bool seq64::event::is_channel_msg (midibyte *msg*) [inline],[static]

Parameters

<i>msg</i>	The channel status or message byte to be tested.
------------	--

Returns

Returns true if the byte represents a MIDI channel message.

9.4.3.3 static bool seq64::event::is_one_byte_msg (midibyte *msg*) [inline],[static]

The rest have two.

Parameters

<i>msg</i>	The channel status or message byte to be tested.
------------	--

Returns

Returns true if the byte represents a MIDI channel message that has only one data byte. However, if this function returns false, it might not be a channel message at all, so be careful.

9.4.3.4 static bool seq64::event::is_two_byte_msg (midibyte *msg*) [inline],[static]

Parameters

<i>msg</i>	The channel status or message byte to be tested.
------------	--

Returns

Returns true if the byte represents a MIDI channel message that has two data bytes. However, if this function returns false, it might not be a channel message at all, so be careful.

9.4.3.5 static bool seq64::event::is_desired_cc_or_not_cc (midibyte *msg*, midibyte *cc*, midibyte *datum*) [inline],[static]

Note

The old logic was the first line, but can be simplified to the second line; the third line shows the abstract representation. Also made sure of this using a couple truth tables.

```
(m != EVENT_CONTROL_CHANGE) || (m == EVENT_CONTROL_CHANGE && d == cc)
(m != EVENT_CONTROL_CHANGE) || (d == cc)
a || (! a && b) => a || b
```

```
\param msg
    The channel status or message byte to be tested.

\param cc
    The desired cc value, which the datum must match, if the message is
    a control-change message.

\param datum
    The current datum, to be compared to cc, if the message is a
    control-change message.

\return
    Returns true if the message is not a control-change, or if it is
    and the cc and datum parameters match.
```

9.4.3.6 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.

9.4.3.7 void seq64::event::set_status (midibyte *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`.

Found in yet another fork of seq24:

```
// ORL fait de la merde
```

He also provided a very similar routine: `set_status_midibus()`.

9.4.3.8 void seq64::event::set_data (midibyte *d1*) [inline]

Parameters

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

9.4.3.9 void seq64::event::set_data (midibyte *d1*, midibyte *d2*) [inline]

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".

9.4.3.10 `void seq64::event::get_data (midibyte & d0, midibyte & d1) const [inline]`

Parameters

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

9.4.3.11 `bool seq64::event::append_sysex (midibyte * data, long dsize)`

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

Parameters

<i>data</i>	Provides the additional SYSEX data. If not provided, nothing is done, and false is returned.
<i>dsize</i>	Provides the size of the additional SYSEX data. If not provided, nothing is done.

Returns

Returns false if there was an `EVENT_SYSEX_END` byte in the appended data, or if an error occurred, and the caller needs to stop trying to process the data.

9.4.3.12 `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.

9.4.4 Field Documentation

9.4.4.1 `midibyte 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.

9.4.4.2 `midibyte seq64::event::m_data[SEQ64_MIDI_DATA_BYTE_COUNT] [private]`

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

9.4.4.3 `midibyte* seq64::event::m_sysex [private]`

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

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

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

9.5 `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.

9.5.1 Detailed Description

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

9.5.2 Constructor & Destructor Documentation

9.5.2.1 `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 <code>event::get_rank()</code> function for more information.

9.5.2.2 `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.
------------	--------------------------------------

9.5.3 Member Function Documentation

9.5.3.1 `bool seq64::event_list::event_key::operator< (const event_key & rhs) const`

Parameters

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

9.5.4 Field Documentation

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

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

9.6 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](#) () const
Counts the selected note-on events in the event list.
- bool [any_selected_notes](#) () const
Indicates that at least one note is selected.
- int [count_selected_events](#) (midibyte status, midibyte cc) const
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.

9.6.1 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 `SEQ64_USE_EVENT_MAP` macro near the top of this module.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 seq64::event_list::event_list (const event_list & rhs)

Parameters

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

9.6.3 Member Function Documentation

9.6.3.1 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.
------------	---

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

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

9.6.3.3 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 <code>std::list</code> implementation has been built in, then the event list is sorted after the addition. This is a time-consuming operation.

9.6.3.4 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 multi-map. 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.

9.6.3.5 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.

9.6.3.6 void seq64::event_list::verify_and_link (long *slength*) [private]

Threadsafe

Parameters

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

9.6.3.7 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.
----------------	---

9.6.3.8 bool seq64::event_list::any_selected_notes () const [private]

Acts like [event_list::count_selected_notes\(\)](#), but stops after finding a selected note. We could add a flag to [count_selected_notes\(\)](#) to break, I suppose.

9.6.3.9 int seq64::event_list::count_selected_events (midibyte *status*, midibyte *cc*) const [private]

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

9.7 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](#) ()
Rote default constructor.
- void [init](#) (Glib::RefPtr< Gdk::Window > windo)
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 > drawable, const char *str, [font::Color](#) col) const
Draws a text string.
- int [char_width](#) () const
'Getter' function for member m_font_w
- int [char_height](#) () const
'Getter' function for member m_font_h
- int [padded_height](#) () const
'Getter' function for member m_padded_h

Private Attributes

- bool [m_use_new_font](#)
If true, use the new font, which is a little bit more modern looking.
- int [m_cell_w](#)
Specifies the cell width of the whole cell.
- int [m_cell_h](#)
Specifies the cell height of the whole cell.
- int [m_font_w](#)
Specifies the exact width of a character cell, in pixels.
- int [m_font_h](#)
Specifies the exact height of a character cell, in pixels.
- int [m_offset](#)
Provides an ad hoc small horizontal or vertical offset for printing strings.
- int [m_padded_h](#)
Provides a common constant used by much of the drawing code, but only marginally related to the padded character height.
- const 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_w.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_b_on_y_pixmap](#)
The pixmap in the file src/pixmaps/font_yb.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.

9.7.1 Member Enumeration Documentation

9.7.1.1 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.

9.7.2 Member Function Documentation

9.7.2.1 void seq64::font::init (Glib::RefPtr< Gdk::Window > wp)

This function loads four pixmaps that contain the characters to be used to draw text strings.

One pixmap has white characters on a black background, one has black characters on a white background, one has yellow characters on a black background, and one has black characters on a yellow background.

9.7.2.2 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) const

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

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

9.7.3 Field Documentation

9.7.3.1 int seq64::font::m_font_w [private]

Currently defaults to cf_text_w = 6. Note that a lot of stuff depends on this being 6 at present, even with our new, slightly wider, font.

9.7.3.2 int seq64::font::m_font_h [private]

Currently defaults to cf_text_h = 10. Note that a lot of stuff depends on this being 10 at present, even with our new, slightly wider, font. But some of the drawing code doesn't use the character height, but the padded character height.

9.7.3.3 `const Glib::RefPtr<Gdk::Pixmap>* seq64::font::m_pixmap` `[mutable], [private]`

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

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

It contains a black font on a white background.

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

It contains a black font on a white background.

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

It contains a black font on a yellow background.

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

It contains a yellow font on a black background.

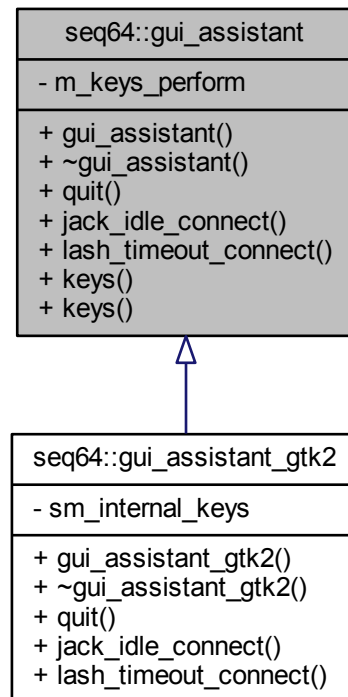
9.7.3.8 `Glib::RefPtr<Gdk::Bitmap> seq64::font::m_clip_mask` `[private]`

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

9.8 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.

9.8.1 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).

9.8.2 Constructor & Destructor Documentation

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

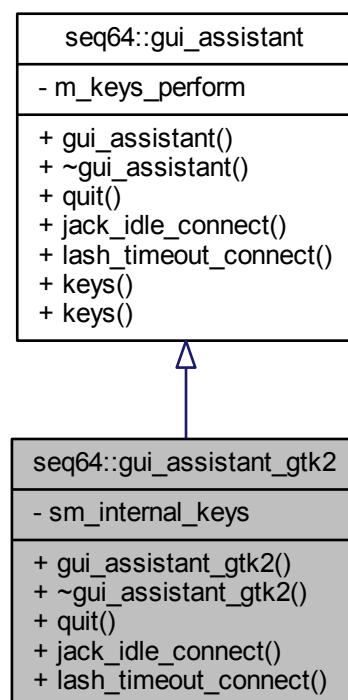
Parameters

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

9.9 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.

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 [clear_window](#) ()
Clears the main window.
- void [set_line](#) (Gdk::LineStyle ls, int width=1)
A small wrapper function for readability in line-drawing.
- void [draw_line](#) (int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on the window.
- void [draw_line](#) (const [Color](#) &c, int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on the window after setting the given foreground color.
- void [draw_line_on_pixmap](#) (int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on the pixmap.
- void [draw_line_on_pixmap](#) (const [Color](#) &c, int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on the pixmap after setting the given foreground color.
- void [draw_line](#) (Glib::RefPtr< Gdk::Pixmap > &pixmap, int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on any pixmap (not a drawable, though, due to a compiler error after setting the given foreground color.
- void [draw_line](#) (Glib::RefPtr< Gdk::Pixmap > &pixmap, const [Color](#) &c, int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on the pixmap after setting the given foreground color.
- void [draw_line](#) (Glib::RefPtr< Gdk::Drawable > &drawable, int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on any pixmap (not a drawable, though, due to a compiler error after setting the given foreground color.
- void [draw_line](#) (Glib::RefPtr< Gdk::Drawable > &drawable, const [Color](#) &c, int x1, int y1, int x2, int y2)
A small wrapper function to draw a line on the drawable after setting the given foreground color.
- void [render_string](#) (int x, int y, const std::string &s, [font::Color](#) color)
A small wrapper function for readability in string-drawing to the window.
- void [render_string_on_pixmap](#) (int x, int y, const std::string &s, [font::Color](#) color)

- A small wrapper function for readability in string-drawing to the pixmap.*

 - void [draw_rectangle](#) (int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on the window.

 - void [draw_rectangle](#) (const [Color](#) &c, int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing.

 - void [draw_rectangle](#) (Glib::RefPtr< Gdk::Drawable > &drawable, int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on a "drawable" context, where the foreground color has already been specified.

 - void [draw_rectangle](#) (Glib::RefPtr< Gdk::Drawable > &drawable, const [Color](#) &c, int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on any drawable context.

 - void [draw_rectangle](#) (Glib::RefPtr< Gdk::Pixmap > &pixmap, int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on a "pixmap" context, where the foreground color has already been specified.

 - void [draw_rectangle](#) (Glib::RefPtr< Gdk::Pixmap > &pixmap, const [Color](#) &c, int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on any pixmap context.

 - void [draw_rectangle_on_pixmap](#) (int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on the pixmap.

 - void [draw_rectangle_on_pixmap](#) (const [Color](#) &c, int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on the pixmap.

 - void [draw_normal_rectangle_on_pixmap](#) (int x, int y, int lx, int ly, bool fill=true)

A small wrapper function for readability in box-drawing on the pixmap.

 - void [draw_drawable](#) (int xsrc, int ysrc, int xdest, int ydest, int width, int height)

Provides the most common use case for redrawing.

 - void [on_realize](#) ()

For this GTK callback, on realization of window, initialize the shiz.

Protected Attributes

- Glib::RefPtr< Gdk::GC > [m_gc](#)

The graphics context, which is required for ever drawing and rendering operation.
- Glib::RefPtr< Gdk::Window > [m_window](#)

Provides the default "window".
- Gtk::Adjustment & [m_vadjust](#)

Provides an object for vertical "adjustments".
- Gtk::Adjustment & [m_hadjust](#)

Provides an object for horizontal "adjustments".
- Glib::RefPtr< Gdk::Pixmap > [m_pixmap](#)

Provides the default "pixmap".
- Glib::RefPtr< Gdk::Pixmap > [m_background](#)

Another pixmap, used for backgrounds.
- Glib::RefPtr< Gdk::Pixmap > [m_foreground](#)

Another pixmap, used for foregrounds.
- [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

9.10.1 Detailed Description

Note that this class really "isn't a" `gui_pallette_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.

9.10.2 Member Function Documentation

9.10.2.1 void `seq64::gui_drawingarea_gtk2::clear_window` () `[inline]`, `[protected]`

One less need to access `m_window` directly.

9.10.2.2 void `seq64::gui_drawingarea_gtk2::set_line` (`Gtk::LineStyle ls`, `int width = 1`) `[inline]`, `[protected]`

Sets the attributes of a line to be drawn.

Parameters

<i>ls</i>	Provides the Gtk-specific line style.
<i>width</i>	Provides the width of the line to be drawn. It defaults to the most common value, 1.

9.10.2.3 void `seq64::gui_drawingarea_gtk2::draw_line` (`int x1`, `int y1`, `int x2`, `int y2`) `[inline]`, `[protected]`

Parameters

<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.4 void `seq64::gui_drawingarea_gtk2::draw_line` (`const Color & c`, `int x1`, `int y1`, `int x2`, `int y2`) `[protected]`

Parameters

<i>c</i>	The foreground color in which to draw the line.
<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.5 void `seq64::gui_drawingarea_gtk2::draw_line_on_pixmap` (`int x1`, `int y1`, `int x2`, `int y2`) `[inline]`, `[protected]`

Parameters

<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.6 `void seq64::gui_drawingarea_gtk2::draw_line_on_pixmap (const Color & c, int x1, int y1, int x2, int y2)`
`[protected]`

Parameters

<i>c</i>	The foreground color in which to draw the line.
<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.7 `void seq64::gui_drawingarea_gtk2::draw_line (Glib::RefPtr< Gdk::Pixmap > & pixmap, int x1, int y1, int x2, int y2)`
`[inline], [protected]`

Parameters

<i>pixmap</i>	Provides the Gdk::Pixmap pointer needed to draw the line.
<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.8 `void seq64::gui_drawingarea_gtk2::draw_line (Glib::RefPtr< Gdk::Pixmap > & pixmap, const Color & c, int x1, int y1, int x2, int y2)` `[protected]`

Parameters

<i>drawable</i>	Provides the Gdk::Drawable pointer needed to draw the line.
<i>c</i>	The foreground color in which to draw the line.
<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.9 `void seq64::gui_drawingarea_gtk2::draw_line (Glib::RefPtr< Gdk::Drawable > & drawable, int x1, int y1, int x2, int y2)` `[inline], [protected]`

Parameters

<i>drawable</i>	Provides the Gdk::Drawable pointer needed to draw the line.
<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.

<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.10 `void seq64::gui_drawingarea_gtk2::draw_line (Glib::RefPtr< Gdk::Drawable > & drawable, const Color & c, int x1, int y1, int x2, int y2)` `[protected]`

Parameters

<i>drawable</i>	Provides the Gdk::Drawable pointer needed to draw the line.
<i>c</i>	The foreground color in which to draw the line.
<i>x1</i>	The x coordinate of the starting point.
<i>y1</i>	The y coordinate of the starting point.
<i>x2</i>	The x coordinate of the ending point.
<i>y2</i>	The y coordinate of the ending point.

9.10.2.11 `void seq64::gui_drawingarea_gtk2::render_string (int x, int y, const std::string & s, font::Color color)` `[inline]`, `[protected]`

Parameters

<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>s</i>	The string to be drawn.
<i>color</i>	The color with which to draw the string.

9.10.2.12 `void seq64::gui_drawingarea_gtk2::render_string_on_pixmap (int x, int y, const std::string & s, font::Color color)` `[inline]`, `[protected]`

Parameters

<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>s</i>	The string to be drawn.
<i>color</i>	The color with which to draw the string.

9.10.2.13 `void seq64::gui_drawingarea_gtk2::draw_rectangle (int x, int y, int lx, int ly, bool fill = true)` `[inline]`, `[protected]`

Parameters

<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.14 `void seq64::gui_drawingarea_gtk2::draw_rectangle (const Color & c, int x, int y, int lx, int ly, bool fill = true)` `[protected]`

It adds setting the foreground color to the [draw_rectangle\(\)](#) function.

Parameters

<i>c</i>	Provides the foreground color to set.
<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.15 `void seq64::gui_drawingarea_gtk2::draw_rectangle (Glib::RefPtr< Gdk::Drawable > & drawable, int x, int y, int lx, int ly, bool fill = true) [inline], [protected]`

Parameters

<i>drawable</i>	The object on which to draw the rectangle.
<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.16 `void seq64::gui_drawingarea_gtk2::draw_rectangle (Glib::RefPtr< Gdk::Drawable > & drawable, const Color & c, int x, int y, int lx, int ly, bool fill = true) [protected]`

It also supports setting the foreground color to the [draw_rectangle\(\)](#) function.

We have a number of such functions: for the main window, for the main pixmap, and for any drawing surface. Is the small bit of conciseness worth it?

Parameters

<i>drawable</i>	The surface on which to draw the box.
<i>c</i>	Provides the foreground color to set.
<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.17 `void seq64::gui_drawingarea_gtk2::draw_rectangle (Glib::RefPtr< Gdk::Pixmap > & pixmap, int x, int y, int lx, int ly, bool fill = true) [inline], [protected]`

Parameters

<i>drawable</i>	The object on which to draw the rectangle.
<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.

<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.18 `void seq64::gui_drawingarea_gtk2::draw_rectangle (Glib::RefPtr< Gdk::Pixmap > & pixmap, const Color & c, int x, int y, int lx, int ly, bool fill = true)` [protected]

It also supports setting the foreground color to the `draw_rectangle()` function.

We have a number of such functions: for the main window, for the main pixmap, and for any drawing surface. Is the small bit of conciseness worth it?

Parameters

<i>pixmap</i>	The surface on which to draw the box.
<i>c</i>	Provides the foreground color to set.
<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.19 `void seq64::gui_drawingarea_gtk2::draw_rectangle_on_pixmap (int x, int y, int lx, int ly, bool fill = true)` [inline], [protected]

Parameters

<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.20 `void seq64::gui_drawingarea_gtk2::draw_rectangle_on_pixmap (const Color & c, int x, int y, int lx, int ly, bool fill = true)` [protected]

It adds setting the foreground color to the `draw_rectangle()` function.

Parameters

<i>c</i>	Provides the foreground color to set.
<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.21 `void seq64::gui_drawingarea_gtk2::draw_normal_rectangle_on_pixmap (int x, int y, int lx, int ly, bool fill = true)` [protected]

It uses Gtk to get the proper background styling for the rectangle.

Parameters

<i>x</i>	The x-coordinate of the origin.
<i>y</i>	The y-coordinate of the origin.
<i>lx</i>	The width of the box.
<i>ly</i>	The height of the box.
<i>fill</i>	If true, fill the rectangle with the current foreground color, as set by <code>m_gc->set_foreground(color)</code> . Defaults to true.

9.10.2.22 `void seq64::gui_drawingarea_gtk2::on_realize ()` [protected]

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

9.10.3 Field Documentation

9.10.3.1 `Glib::RefPtr<Gdk::Window> seq64::gui_drawingarea_gtk2::m_window` [protected]

Wrapper functions with undecorated wrapper names are used for accessing this item. We hope to be able to hide this items completely some day.

9.10.3.2 `Glib::RefPtr<Gdk::Pixmap> seq64::gui_drawingarea_gtk2::m_pixmap` [protected]

Wrapper functions with undecorated wrapper names are used for accessing this item. We hope to be able to hide this items completely some day.

9.10.3.3 `Glib::RefPtr<Gdk::Pixmap> seq64::gui_drawingarea_gtk2::m_background` [protected]

Our wrappers still leave this member exposed <giggle>.

9.10.3.4 `Glib::RefPtr<Gdk::Pixmap> seq64::gui_drawingarea_gtk2::m_foreground` [protected]

Our wrappers still leave this member exposed.

9.10.3.5 `perform& seq64::gui_drawingarea_gtk2::m_mainperf` [protected]

We could move this into yet another base class, since a number of classes don't need it. Probably not worth the effort at this time.

9.10.3.6 `int seq64::gui_drawingarea_gtk2::m_window_x` [protected]

Could make this constant, but some windows are resizable.

9.10.3.7 `int seq64::gui_drawingarea_gtk2::m_drop_x` [protected]

Provides the x and y value of where the dragging started.

9.11 seq64::gui_palette_gtk2 Class Reference

Implements a stock palette of `Gdk::Color` elements.

[illegible]

- `gui_palette_gtk2 ()`
Principal constructor.
- `~gui_palette_gtk2 ()`
Provides a destructor to delete allocated objects.

- typedef Gdk::Color **Color**
Provides a type for the color object.

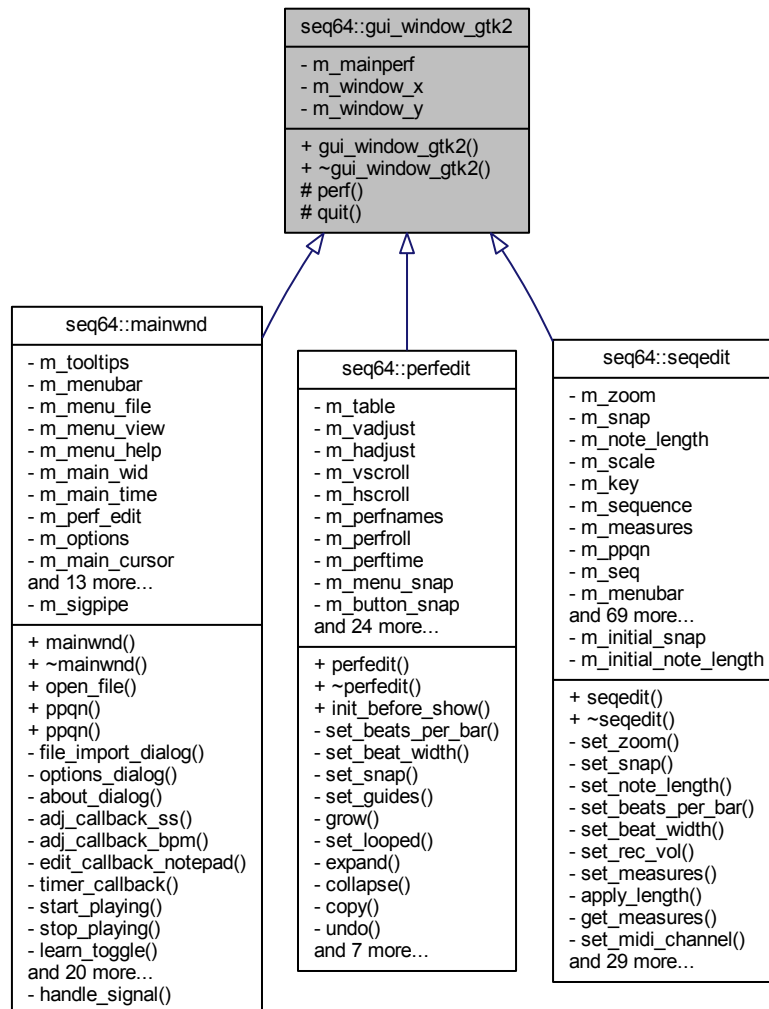
Note that this class must be derived from `Gtk::DrawingArea` (or `Gtk::Widget`) in order to get access to the `get_←`
default `colormap()` function used in the constructor.

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

9.12 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.

Protected Member Functions

- [perform](#) & [perf](#) ()
'Getter' function for member m_mainperf

Private Attributes

- `perform` & `m_mainperf`
The master object, sort of a sequence buss.
- `int m_window_x`
Window sizes.

9.12.1 Constructor & Destructor Documentation

9.12.1.1 `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.
---------------------	--

9.12.2 Field Documentation

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

Could make this constant, but some windows are resizable.

9.13 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.

Private Member Functions

- void [info_message](#) (const std::string &msg)
Common-code for console messages.
- void [error_message](#) (const std::string &msg)
Common-code for error messages.

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)
This callback is to shutdown JACK by clearing the jack_assistant::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.

9.13.1 Constructor & Destructor Documentation

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

Parameters

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

9.13.2 Member Function Documentation

9.13.2.1 `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.)

9.13.2.2 `void seq64::jack_assistant::stop ()`

Should it also set m_jack_running to false?

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

<http://jackaudio.org/files/docs/html/transport-design.html>

This function is called via [perform::position_jack\(\)](#) in the mainwnd, perfedit, perfroll, and seqroll graphical user-interface support objects.

Warning

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

Parameters

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

9.13.2.4 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.

9.13.2.5 void seq64::jack_assistant::info_message (const std::string & msg) [private]

Adds markers and a newline.

Parameters

<i>msg</i>	The message to print, sans the newline.
------------	---

9.13.2.6 void seq64::jack_assistant::error_message (const std::string & msg) [private]

Adds markers, and sets m_jack_running to false.

Parameters

<i>msg</i>	The message to print, sans the newline.
------------	---

9.13.3 Friends And Related Function Documentation

9.13.3.1 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 jack_assistant object. Currently not checked for nullity, nor dynamic-casted.

9.13.3.2 void jack_shutdown (void * arg) [friend]

Parameters

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

9.13.3.3 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.

9.14 seq64::jack_scratchpad Struct Reference

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

9.14.1 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\(\)](#).

9.15 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???

9.15.1 Member Enumeration Documentation

9.15.1.1 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.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 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.

9.15.3 Member Function Documentation

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

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

9.15.3.2 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`.

9.15.4 Field Documentation

9.15.4.1 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.

9.16 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*
- bool [show_ui_sequence_number](#) () const
Accessor *m_key_show_ui_sequency_number*
- 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

- bool [m_key_show_ui_sequence_key](#)
If set, shows the shortcut-keys on each filled pattern slot in the main window.
- bool [m_key_show_ui_sequence_number](#)
If set, shows the sequence number on each filled pattern and empty pattern slot in the main window.
- unsigned int [m_key_bpm_up](#)
Provides key assignments for some key sequencer features.

9.16.1 Detailed Description

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

9.16.2 Constructor & Destructor Documentation

9.16.2.1 seq64::keys_perform::~~keys_perform ()

Finally, any active patterns/sequences are deleted.

9.16.3 Member Function Documentation

9.16.3.1 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.
------------	--

9.16.3.2 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.
------------	--

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

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

9.16.3.4 bool seq64::keys_perform::show_ui_sequence_number () const [inline]

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

9.16.3.5 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](#).

9.16.3.6 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](#).

9.16.3.7 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](#).

9.16.3.8 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.

9.16.3.9 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.

9.16.4 Field Documentation

9.16.4.1 bool seq64::keys_perform::m_key_show_ui_sequence_number [private]

Also show the sequence number as part of the sequence name in the performance window (song editor).

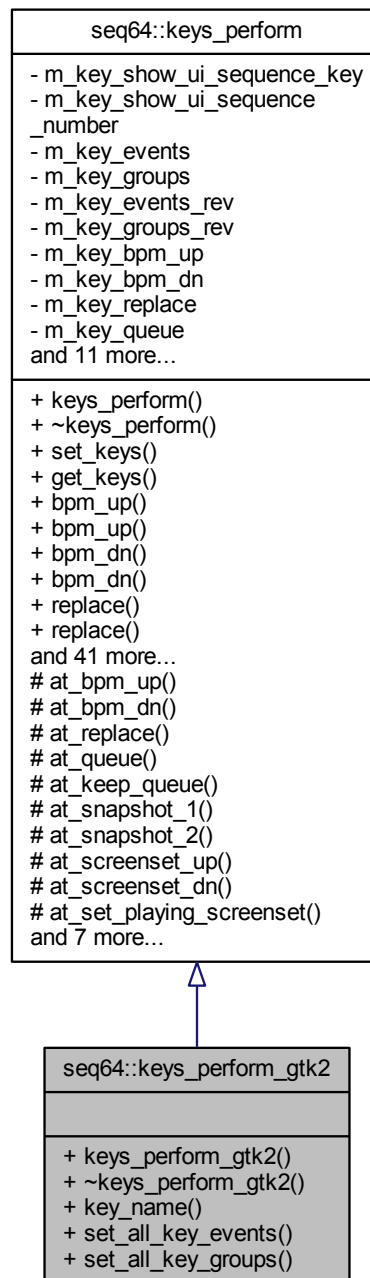
9.16.4.2 unsigned int seq64::keys_perform::m_key_bpm_up [private]

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

9.17 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

9.17.1 Detailed Description

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

9.17.2 Constructor & Destructor Documentation

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

Finally, any active patterns/sequences are deleted.

9.17.3 Member Function Documentation

9.17.3.1 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](#).

9.17.3.2 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](#).

9.17.3.3 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](#).

9.18 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().

9.19 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.

9.19.1 Detailed Description

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

9.19.2 Constructor & Destructor Documentation

- 9.19.2.1 [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.

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

Parameters

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

9.19.3 Member Function Documentation

9.19.3.1 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.

9.19.3.2 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

9.19.4 Field Documentation

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

See the SEQ64_KEYSTROKE_PRESS and SEQ64_KEYSTROKE_RELEASE readability macros.

9.19.4.2 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.

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

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

9.20 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.

9.20.1 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.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 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.

9.20.3 Member Function Documentation

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

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

9.20.3.2 bool seq64::lash::process_events ()

Returns

Always returns true.

9.20.3.3 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.

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

Parameters

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

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

Currently incomplete.

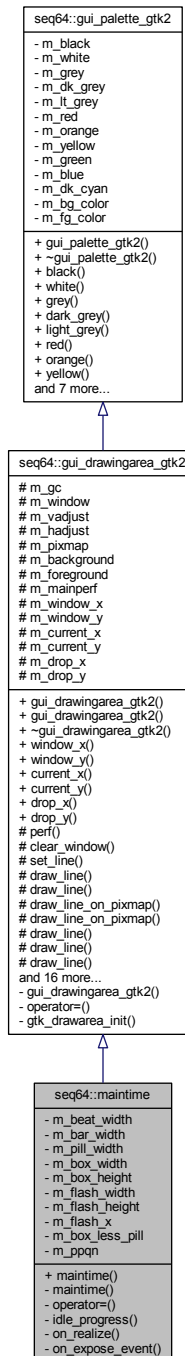
Parameters

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

9.21 seq64::maintime Class Reference

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

Inheritance diagram for seq64::maintime:



Public Member Functions

- [maintime](#) ([perform](#) &p, int ppqn=SEQ64_USE_DEFAULT_PPQN)

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

Private Member Functions

- 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.
- void [on_realize](#) ()
Handles realization of the window.
- bool [on_expose_event](#) (GdkEventExpose *ev)
This function merely idles.

Private Attributes

- const int [m_beat_width](#)
Provides the divisor for ticks to produce a beat value.
- const int [m_bar_width](#)
Provides the divisor for ticks to produce a bar value.
- const int [m_pill_width](#)
Provides the width of the pills, little black squares that show the progress of a beat and a bar (measure).
- const int [m_box_width](#)
The width/length of the rectangle to be drawn inside the maintime window.
- const int [m_box_height](#)
The height of the rectangle to be drawn inside the maintime window.
- const int [m_flash_width](#)
The width/length of the flashing rectangle to be drawn inside the maintime window.
- const int [m_flash_height](#)
The height of the flashing rectangle to be drawn inside the maintime window.
- const int [m_flash_x](#)
The x value at which a flash should occur.
- const int [m_box_less_pill](#)
The width/length of the maintime window minus the width of the pill.
- int [m_ppqn](#)
Provides the active PPQN value.

Additional Inherited Members

9.21.1 Detailed Description

We added a lot of members to hold the results of calculations that involve what are essentially constant. This saves CPU time, and maybe a little memory for the code to make those calculations more than once.

9.21.2 Constructor & Destructor Documentation

9.21.2.1 seq64::maintime::maintime (**perform** & *p*, int *ppqn* = SEQ64_USE_DEFAULT_PPQN)

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

9.21.3 Member Function Documentation

9.21.3.1 `int seq64::maintime::idle_progress (long ticks) [private]`

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".

Parameters

<i>ticks</i>	Provides the main tick setting. This setting is provided by mainwnd(), in its timer callback.
--------------	---

Returns

Always returns 1 (it used to return "true!").

9.21.3.2 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.

9.21.3.3 bool seq64::maintime::on_expose_event (GdkEventExpose * a_e) [private]

We don't need the m_tick member, the function works as well if 0 is passed in. We've removed m_tick permanently.

9.21.4 Field Documentation

9.21.4.1 const int seq64::maintime::m_beat_width [private]

Currently, this value is hardwired to 4, but will eventually be wired up as usr().midi_beat_width().

9.21.4.2 const int seq64::maintime::m_bar_width [private]

Currently, this value is hardwired to 16, but will eventually be wired up as usr().midi_beat_width() * usr().midi_beats_per_bar().

9.21.4.3 const int seq64::maintime::m_box_width [private]

This item absolutely depends on the main window being non-resizable.

9.21.4.4 const int seq64::maintime::m_box_height [private]

This item absolutely depends on the main window being non-resizable.

9.21.4.5 const int seq64::maintime::m_flash_width [private]

Just a bit smaller than m_box_width.

9.21.4.6 const int seq64::maintime::m_flash_height [private]

Just a bit smaller than m_box_width.

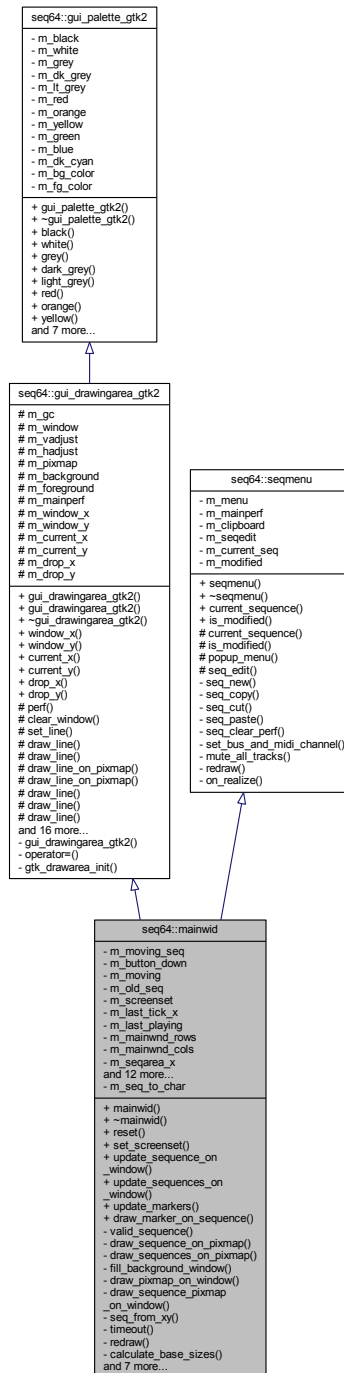
9.21.4.7 int seq64::maintime::m_ppqn [private]

While this is effectively a constant for the duration of a tune, it might change as different tunes are loaded.

9.22 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.
- int [m_screenset_slots](#)
Provides a convenience variable for avoiding multiplications.
- int [m_screenset_offset](#)
Provides a convenience variable for avoiding multiplications.

Additional Inherited Members

9.22.1 Constructor & Destructor Documentation

9.22.1.1 `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.
----------------	--

9.22.2 Member Function Documentation

9.22.2.1 `void seq64::mainwid::set_screenset (int ss)`

Parameters

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

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

Parameters

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

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

Parameters

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

9.22.2.4 `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

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

9.22.2.5 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.

9.22.2.6 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 play – the progress bars move in each pattern. Gaps in the sequence in the Song (performance) Editor. don't change the appearance of the patterns. But, if the Song Editor window is up, and the song is started using the controls in the Song Editor, then the active patterns are black while playing, and white when gaps in the sequence are encountered. The muting status in the main window is ignored. The muting in the Song (performance) windows is in force.

Parameters

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

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

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

9.22.2.8 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.
---------------	--

9.22.2.9 int seq64::mainwid::seq_from_xy (int *x*, int *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.

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

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

Returns

Always returns true.

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

Parameters

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

Implements [seq64::seqmenu](#).

9.22.2.12 `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.

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

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

This function used to call [font::init\(\)](#), and was the only place where the [font::init\(\)](#) function was called. The [init\(\)](#) function gets a color-map from the window. We need a more fool-proof way to do this!

9.22.2.14 `bool seq64::mainwid::on_expose_event (GdkEventExpose * ev) [private]`

Parameters

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

Returns

Always returns true.

9.22.2.15 `bool seq64::mainwid::on_button_press_event (GdkEventButton * p)` `[private]`

If the press is a single left-click, and no Ctrl key is pressed, then this function 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. In the release event callback, this then causes the sequence arming/muting to be toggled.

If the press is a single Ctrl-left-click, this function brings up the New or Edit menu. The New menu is brought up if the grid slot is empty, and the Edit menu otherwise.

If the press is a double-click, it first acts just like two single-clicks (which might confuse the user at first). Then it brings up the Edit menu for the sequence. This new behavior is closer to what users have come to expect from a double-click.

Parameters

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

Returns

Always returns true.

9.22.2.16 `bool seq64::mainwid::on_button_release_event (GdkEventButton * p)` `[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>p</i>	Provides the parameters of the button event.
----------	--

Returns

Always returns true.

9.22.2.17 `bool seq64::mainwid::on_motion_notify_event (GdkEventMotion * p)` `[private]`

This function moves the selected pattern to another pattern slot.

The [perform::delete_sequence\(\)](#) function sets the perform modification flag.

Parameters

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

Returns

Always returns true.

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

Just sets the `Gtk::HAS_FOCUS` flag.

Returns

Always returns false.

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

Just unsets the Gtk::HAS_FOCUS flag.

Returns

Always returns false.

9.22.3 Field Documentation

9.22.3.1 `int seq64::mainwid::m_screenset_slots [private]`

It is equally to `m_mainwnd_rows * m_mainwnd_cols`.

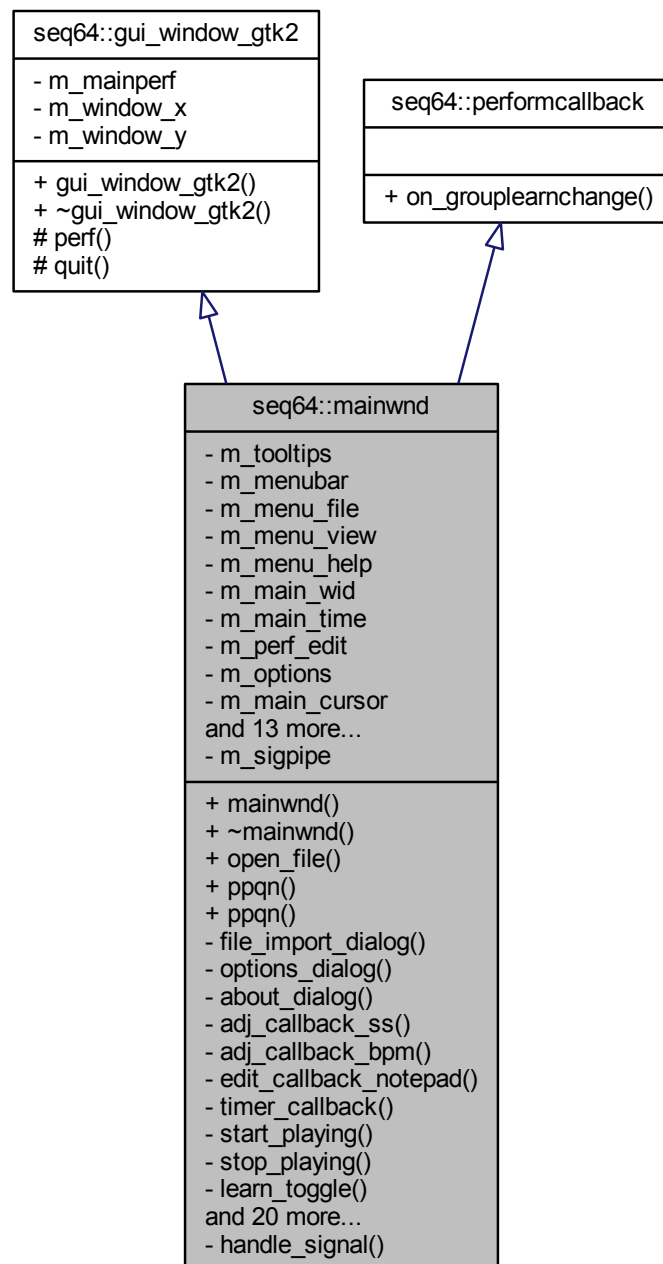
9.22.3.2 `int seq64::mainwid::m_screenset_offset [private]`

It is equally to `m_screenset_slots * m_screenset`.

9.23 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, the global ppqn, if SEQ64_USE_DEFAULT↔T_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

9.23.1 Constructor & Destructor Documentation

9.23.1.1 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.

9.23.2 Member Function Documentation

9.23.2.1 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 the global 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 "oldformat" or "global sequence" parameters of the midifile constructor when reading the MIDI file, since reading handles both the old and new formats, dealing with new constructs only if they are present in the file.

Parameters

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

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

```
m_ppqn = choose_ppqn(ppqn);
```

9.23.2.3 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, as well as being able to offset the sequence number.

Also, it is important to note that [perf\(\).clear_all\(\)](#) is not called by this routine, as we are merely adding to what might already be there.

9.23.2.4 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.

9.23.2.5 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.

Let the perform object keep track of modifications.

Screen-set notepad?

9.23.2.6 void seq64::mainwnd::adj_callback_bpm () [private]

Let the perform object keep track of modifications.

9.23.2.7 void seq64::mainwnd::edit_callback_notepad () [private]

Let the perform object keep track of modifications.

9.23.2.8 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.

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

We will let perform keep track of modifications, and not just set an is-modified flag just because we opened the song editor. We're going to centralize the modification flag in the perform object, and see if it can work.

Todo Can we offload all this work to perfedit? Is it worthwhile?

9.23.2.10 `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.

9.23.2.11 `void seq64::mainwnd::new_file () [private]`

Not sure that we need to clear the modified flag here, especially since it is now centralized in the perform object. Let `perf().clear_all()` handle it now.

9.23.2.12 `bool seq64::mainwnd::save_file () [private]`

Here we specify the current value of `m_ppqn`, which was set when reading the MIDI file. We also let `midifile` tell the perform that saving worked, so that the "is modified" flag can be cleared. The `midifile` class is already a friend of `perform`.

9.23.2.13 `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.

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

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

9.23.2.15 `bool seq64::mainwnd::on_key_press_event (GdkEventKey * 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.

9.23.2.16 `bool seq64::mainwnd::on_key_release_event (GdkEventKey * 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.

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

This handler responds to a learn-mode change from `perf()`.

Reimplemented from `seq64::performcallback`.

9.23.3 Field Documentation

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

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

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

The first is the Patterns Panel.

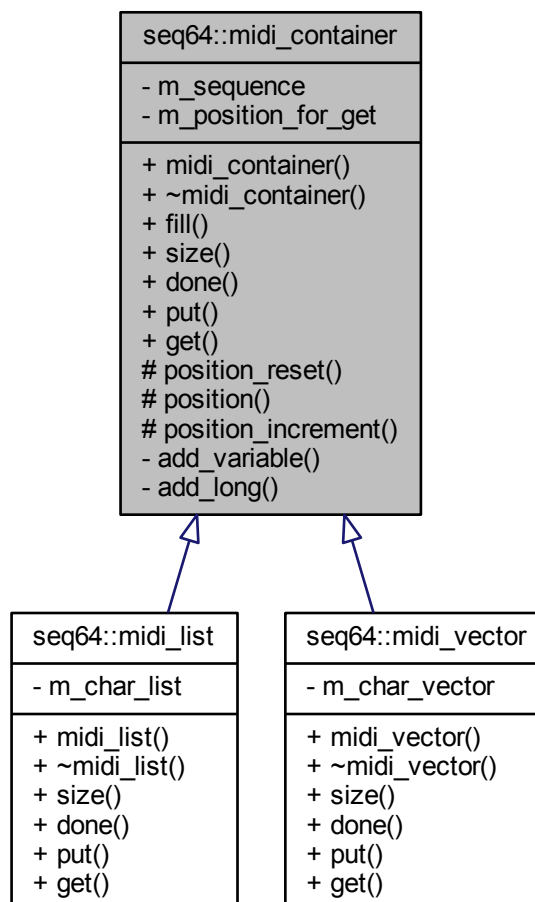
9.23.3.3 `Gtk::SpinButton* seq64::mainwnd::m_spinbutton_load_offset` `[private]`

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

9.24 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 track (sequence) 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.

9.24.1 Member Function Documentation

9.24.1.1 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 use with that shortcut.

Triggers:

Triggers are added by first calling `add_variable(0)`, which is needed because why?

Then `0xFF 0x7F` is written, followed by the length value, which is the number of triggers at 3 long integers per trigger, plus the 4-byte code for triggers, `c_triggers_new = 0x24240008`.

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.
--------------------	--

9.24.1.2 `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](#).

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

It also increments `m_position_for_get`.

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

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

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

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

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

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

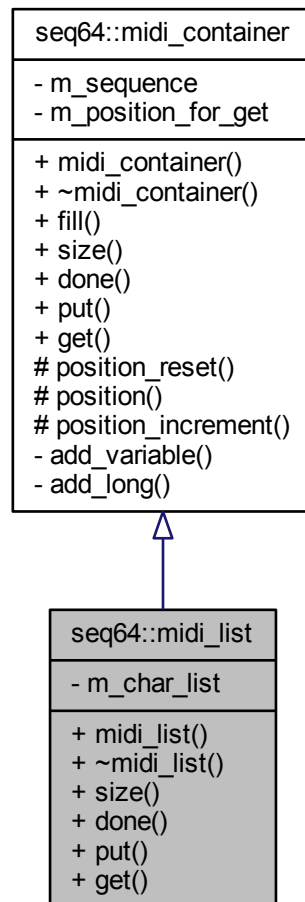
This function "replaces" `sequence::add_long_list()`.

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.

9.25 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

9.25.1 Member Typedef Documentation

9.25.1.1 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.

9.25.2 Member Function Documentation

9.25.2.1 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](#).

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

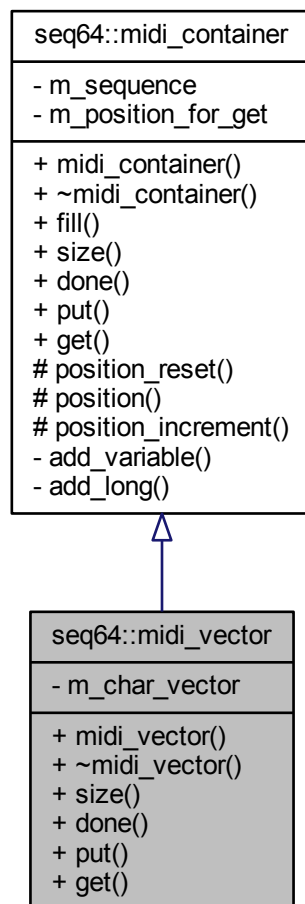
In this implementation, m_position_for_get is not used. The elements of the container are popped off backward!

Implements [seq64::midi_container](#).

9.26 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

9.26.1 Member Function Documentation

9.26.1.1 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](#).

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

In this implementation, m_position_for_get is used.

Implements [seq64::midi_container](#).

9.27 seq64::midibus Class Reference

Provides a class for handling the MIDI buss on Linux.

Public Member Functions

- [~midibus \(\)](#)
A rote empty destructor.
- bool [init_out \(\)](#)
Initialize the MIDI output port.
- bool [init_in \(\)](#)
Initialize the MIDI input port.
- bool [deinit_in \(\)](#)
Deinitialize the MIDI input?
- bool [init_out_sub \(\)](#)
Initialize the output in a different way?
- bool [init_in_sub \(\)](#)
Initialize the output in a different way?
- void [print \(\)](#)
Prints m_name.
- const std::string & [get_name \(\)](#) const
'Getter' function for member n_name
- int [get_id \(\)](#) const
'Getter' function for member m_id
- void [play \(event *e24, midibyte channel\)](#)

- This `play()` function takes a native event, encodes it to ALSA event, and puts it in the queue.*

 - void `sysex` (`event *e24`)

Takes a native SYSEX event, encodes it to an ALSA event, and then puts it in the queue.
- void `start` ()

This function gets the MIDI clock a-runnin', if the clock type is not `e_clock_off`.
- void `stop` ()

Stop the MIDI buss.
- void `clock` (long tick)

Generates the MIDI clock, starting at the given tick value.
- void `continue_from` (long tick)

Continue from the given tick.
- void `init_clock` (long tick)

Initialize the clock, continuing from the given tick.
- void `set_clock` (`clock_e` clocktype)

'Setter' function for member `m_clock_type`
- `clock_e` `get_clock` () const

'Getter' function for member `m_clock_type`
- void `set_input` (bool inputing)

Input functions.
- bool `get_input` () const

'Getter' function for member `m_inputing`
- void `flush` ()

Flushes our local queue events out into ALSA.
- int `get_client` () const

'Getter' function for member `m_dest_addr_client` The address of client.
- int `get_port` () const

'Getter' function for member `m_dest_addr_port`

Static Public Member Functions

- static void `set_clock_mod` (int clockmod)

Set the clock mod to the given value, if legal.
- static int `get_clock_mod` ()

Get the clock mod.

Private Attributes

- int `m_id`

The ID of the midibus object.
- `clock_e` `m_clock_type`

The type of clock to use.
- bool `m_inputing`

TBD.
- int `m_ppqn`

Provides the PPQN value in force, currently a constant.
- const int `m_dest_addr_client`

Destination address of client.
- const int `m_dest_addr_port`

Destination port of client.
- const int `m_local_addr_client`

- Local address of client.*
- int [m_local_addr_port](#)
Local port of client.
- int [m_queue](#)
Another ID of the MIDI queue?
- std::string [m_name](#)
The name of the MIDI buss.
- long [m_lasttick](#)
The last (most recent? final?) tick.
- mutex [m_mutex](#)
Locking mutex.

Static Private Attributes

- static int [m_clock_mod](#)
*This is another name for "16 * 4".*

Friends

- class [mastermidibus](#)
The master MIDI bus sets up the buss.

9.27.1 Member Function Documentation

9.27.1.1 `bool seq64::midibus::init_out ()`

Returns

Returns true unless setting up ALSA MIDI failed in some way.

9.27.1.2 `bool seq64::midibus::init_in ()`

Returns

Returns true unless setting up ALSA MIDI failed in some way.

9.27.1.3 `bool seq64::midibus::deinit_in ()`

Returns

Returns true, unless an error occurs.

9.27.1.4 `bool seq64::midibus::init_out_sub ()`

Returns

Returns true unless setting up the ALSA port failed in some way.

9.27.1.5 `bool seq64::midibus::init_in_sub ()`

Returns

Returns true unless setting up the ALSA port failed in some way.

9.27.1.6 `void seq64::midibus::play (event * e24, midibyte channel)`

Threadsafe

Parameters

<i>e24</i>	The event to be played on this bus.
<i>channel</i>	The channel of the playback.

9.27.1.7 `void seq64::midibus::sysex (event * e24)`

Parameters

<i>e24</i>	The event to be handled.
------------	--------------------------

9.27.1.8 `void seq64::midibus::clock (long tick)`

Parameters

<i>tick</i>	Provides the starting tick.
-------------	-----------------------------

9.27.1.9 `void seq64::midibus::continue_from (long tick)`

Parameters

<i>tick</i>	The continuing tick.
-------------	----------------------

9.27.1.10 `void seq64::midibus::init_clock (long tick)`

Parameters

<i>tick</i>	The starting tick.
-------------	--------------------

9.27.1.11 `void seq64::midibus::set_input (bool inputing)`

Set status to of "inputting" to the given value.

If the parameter is true, then [init_in\(\)](#) is called; otherwise, [deinit_in\(\)](#) is called.

Parameters

<i>inputing</i>	The inputing value to set.
-----------------	----------------------------

9.28 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 oldformat=false, bool globalbgs=true)
Principal constructor.
- [~midifile](#) ()
A rote destructor.
- bool [parse](#) ([perform](#) &a_perf, int a_screen_set=0)
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.
- const std::string & [error_message](#) () const
'Getter' function for member m_error_message
- 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 Types

- typedef std::map< int, [sequence](#) * > [ChannelMap](#)
Provides SMF 0 support for breaking a multi-channel SMF 0 file into a number of sequences.

Private Member Functions

- void [init_smf0_support](#) ()
Resets the SMF 0 support variables in preparation for parsing a new MIDI file.
- void [smf0_increment](#) (int channel)
Processes a channel number by raising its flag in the m_smf0_channels[] array.
- bool [parse_smf_0](#) ([perform](#) &p, int screenset)
This function parses an SMF 0 binary MIDI file as if it were an SMF 1 file, then, if more than one MIDI channel was encountered in the sequence, splits all of the channels in the sequence out separate sequences, and deletes the original sequence.
- bool [parse_smf_1](#) ([perform](#) &p, int screenset)
This function parses an SMF 1 binary MIDI file; it is basically the original seq25 [midifile::parse\(\)](#) function.
- 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\(\)](#).
- midibyte [read_byte](#) ()
Reads 1 byte of data directly from 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 `read_byte_array` (midibyte *b, int len)
A helper function to simplify reading midi_control data from the MIDI file.
- void `write_byte` (midibyte 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.
- std::string `read_track_name` ()
Reads the track name.
- void `write_seq_number` (unsigned short seqnum)
Writes out a sequence number.
- int `read_seq_number` ()
Reads the 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.
- void `errdump` (const std::string &msg)
Helper function to emit more useful error messages.
- void `errdump` (const std::string &msg, unsigned long p)
Helper function to emit more useful error messages for erroneous long values.
- 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.
- bool `is_sysex_special_id` (midibyte ch)
Check for special SysEx ID byte.

Private Attributes

- int `m_file_size`
Holds the size of the MIDI file.
- std::string `m_error_message`
Holds the last error message, useful for trouble-shooting without having Sequencer64 running in a console window.
- bool `m_disable_reported`
Indicates that file reading has already been disabled (due to serious errors), so don't complain about it anymore.
- int `m_pos`
Holds the position in the MIDI file.
- const std::string `m_name`
The unchanging name of the MIDI file.

- `std::vector< midibyte > m_data`
This vector of characters holds our MIDI data.
- `std::list< midibyte > 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.
- `bool m_global_bgsequence`
Indicates to store the new key, scale, and background sequence in the global, "proprietary" section of the MIDI song.
- `int m_ppqn`
Provides the current value of the PPQN, which used to be constant and is now only the macro `DEFAULT_PPQN`.
- `bool m_use_default_ppqn`
Indicates that the default PPQN is in force.
- `int m_smf0_channels_count`
Provides support for SMF 0, indicates how many channels were found in the file in a single sequence.
- `bool m_smf0_channels[16]`
Provides support for SMF 0, holds a bool value that indicates the occurrence of a given channel.
- `ChannelMap m_smf0_map`
Provides support for SMF 0, holds a map of sequences keyed by channel number.

9.28.1 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.

9.28.2 Constructor & Destructor Documentation

9.28.2.1 `seq64::midifile::midifile (const std::string & name, int ppqn = SEQ64_USE_DEFAULT_PPQN, bool oldformat = false, bool globalbgs = true)`

Parameters

<i>name</i>	Provides the name of the MIDI file to be read or written.
<i>ppqn</i>	<p>Provides the initial value of the PPQN setting. It is handled differently for parsing (reading) versus writing the MIDI file.</p> <ul style="list-style-type: none"> • Reading. <ul style="list-style-type: none"> – If set to <code>SEQ64_USE_DEFAULT_PPQN</code>, the legacy application behavior is used. The <code>m_ppqn</code> member is set to the default PPQN, <code>DEFAULT_PPQN</code>. The value read from the MIDI file, <code>ppqn</code>, is then use to scale the running-time of the sequence relative to <code>DEFAULT_PPQN</code>. – Otherwise, <code>m_ppqn</code> is set to the value read from the MIDI file. No scaling is done. Since the value gets written, specify <code>ppqn</code> as 0, an obviously bogus value, to get this behavior. • Writing. This value is written to the MIDI file in the header chunk of the song. Note that the caller must query for the PPQN set during parsing, and pass it to the constructor when preparing to write the file. See how it is done in the <code>mainwnd</code> class.

9.28.3.2 `bool seq64::midifile::write (perform & p)`

Parameters

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

Returns

Returns true if the write operations succeeded.

Note

Seq24 reverses the order of some events, due to popping from its container. Not an issue here.

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

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

9.28.3.4 `void seq64::midifile::smf0_increment (int channel) [private]`

If it is the first entry for that channel, m_smf0_channels_count is incremented. We won't check the channel number, to save time, until someday we segfault :-D

9.28.3.5 `bool seq64::midifile::parse_smf_0 (perform & p, int screenset) [private]`**Parameters**

<i>p</i>	Provides a reference to the perform object into which sequences/tracks are to be added.
<i>screenset</i>	The screen-set offset to be used when loading a sequence (track) from the file.

Returns

Returns true if the parsing succeeded.

9.28.3.6 `bool seq64::midifile::parse_smf_1 (perform & p, int screenset) [private]`

It assumes the file-data has already been read into memory. It also assumes that the ID, track-length, and format have already been read.

Parameters

<i>p</i>	Provides a reference to the perform object into which sequences/tracks are to be added.
<i>screenset</i>	The screen-set offset to be used when loading a sequence (track) from the file.

Returns

Returns true if the parsing succeeded.

9.28.3.7 `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:

```
0x00 0xFF 0x7F length data
```

For convenience, this function first checks the amount of file data left. If enough, then it reads a long value. If the value starts with 0x00 0xFF 0x7F, then that is a SeqSpec event, which signals usage of the new Sequencer64 "proprietary" 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.

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

It consists of series of tags:

- `c_midictrl`
- `c_midiclocks`
- `c_notes`
- `c_bpmtag` (beats per minute)
- `c_mutegroups`
- `c_musickey` (new, added if `usr() global_seq_feature()` is true)
- `c_musicscale` (ditto)
- `c_backsequence` (ditto)

(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. Oh, it turns out that they are set up by actions performed on each sequence, and are stored as sequencer-specific ("SeqSpec") data with each track's data as held in the MIDI container for the track. See the [midi_container](#) module for more information.)

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

First, we separate out this function for a little more clarity. Then we added code to handle reading both the legacy Seq24 format and the new, MIDI-compliant format. Note that even the new 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. But it does have the "MTrk" marker and track name, so that must be processed for the new format.

Now, in our "midicvt" project, we have a test MIDI file, `b4uacuse-non-mtrk.midi` that is good, except for having a tag "MUnk" instead of "MTrk". We should consider being more permissive, if possible. Otherwise, though, the only penalty is that the "proprietary" chunk is completely skipped.

Parameters

<i>p</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 are also implicit parameters, with the `m_pos` and `m_new_format` member variables.

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

Warning

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

9.28.3.10 `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.

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

Warning

This code looks endian-dependent.

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

Warning

This code looks endian-dependent.

9.28.3.13 `void seq64::midifile::read_byte_array (midibyte * b, int len) [inline], [private]`

Parameters

<i>b</i>	The byte array to receive the data.
<i>len</i>	The number of bytes in the array, and to be read.

9.28.3.14 `void seq64::midifile::write_byte (midibyte c) [inline], [private]`

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

9.28.3.15 `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.

9.28.3.16 `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.

9.28.3.17 `std::string seq64::midifile::read_track_name () [private]`

Meant only for usage in the proprietary footer track, in the new file format.

Returns

Returns the track name, or an empty string if there was a problem.

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

The format is "00 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. Our application doesn't bother with that shortcut.

9.28.3.19 `int seq64::midifile::read_seq_number () [private]`

Meant only for usage in the proprietary footer track, in the new file format.

Returns

Returns the sequence number found, or -1 if it was not found.

9.28.3.20 `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. Sequence number, a special value, well out of normal range.
- The name of the track:
 - "Seq24-Spec"

– "Sequencer64-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. Similary, this site shows that no manufacturer uses 0x24:

<http://sequence15.blogspot.com/2008/12/midi-manufacturer-ids.html>

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 c_midibus or c_mutegroups.
<i>data_length</i>	The amount of data that will be written. This parameter does not count the length of the header itself.

9.28.3.21 bool seq64::midifile::write_proprietary_track (perform & p) [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.

Here's the basics of what Seq24 did for writing the data in this part of the file:

```
-# Write the c_midictrl value, then write a 0. To us, this looks like
no one wrote any code to write this data. And yet, the parsing
code can handles a non-zero value, which is the number of sequences
as a long value, not a byte. So shouldn't we write 4 bytes, not
one? Yes, indeed, we made a mistake. However, we should be
writing out the full data set as well. But not even Seq24 does
that! Perhaps they decided it was best kept in the "rc"
configuration file.
-# MORE TO COME.
```

9.28.3.22 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.

9.28.3.23 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 then the data length is added.

9.28.3.24 void seq64::midifile::errdump (const std::string & msg) [private]

It adds the file offset to the message.

Parameters

<i>msg</i>	The main error message string, without an ending newline character.
------------	---

Returns

The constructed string is returned as a side-effect, in case we want to pass it along to the externally-accessible error-message buffer.

9.28.3.25 void seq64::midifile::errdump (const std::string & msg, unsigned long value) [private]

It adds the file offset to the message.

Parameters

<i>msg</i>	The main error message string, without an ending newline character.
<i>value</i>	The long value to show as part of the message.

Returns

The constructed string is returned as a side-effect, in case we want to pass it along to the externally-accessible error-message buffer.

9.28.3.26 bool seq64::midifile::is_sysex_special_id (midibyte ch) [inline], [private]**Parameters**

<i>ch</i>	Provides the byte to be checked against 0x7D through 0x7F.
-----------	--

Returns

Returns true if the byte is SysEx special ID.

9.28.4 Field Documentation

9.28.4.1 `int seq64::midifile::m_file_size` [private]

This variable was added when loading a file that caused an attempt to load data well beyond the file-size of the midicvt test file Dixie04.mid.

9.28.4.2 `std::string seq64::midifile::m_error_message` [private]

If empty, there's no pending error. Currently most useful in the [parse\(\)](#) function.

9.28.4.3 `bool seq64::midifile::m_disable_reported` [private]

Once is enough.

9.28.4.4 `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.

9.28.4.5 `std::vector<midibyte> 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.

9.28.4.6 `std::list<midibyte> 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.

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

In the 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, the new format is used, 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 "rc" configuration file.] Note that reading can handle either format transparently.

9.28.4.8 `int seq64::midifile::m_smf0_channels_count` [private]

SMF 1 file parsing will only warn about more than one channel found in a given sequence.

9.28.4.9 `bool seq64::midifile::m_smf0_channels[16]` [private]

Obviously, we don't have to worry about multiple MIDI busses.

9.29 `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.

9.29.1 Field Documentation

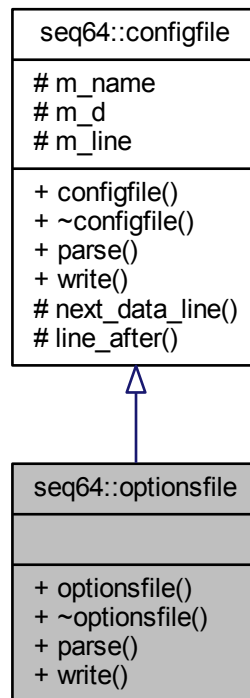
9.29.1.1 Gtk::Notebook* seq64::options::m_notebook [private]

Must be a GTK thang.

9.30 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

9.30.1 Detailed Description

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

9.30.2 Member Function Documentation

9.30.2.1 bool seq64::optionsfile::parse (perform & p) [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](#).

9.30.2.2 `bool seq64::optionsfile::write (const perform & p) [virtual]`

Parameters

<i>p</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.

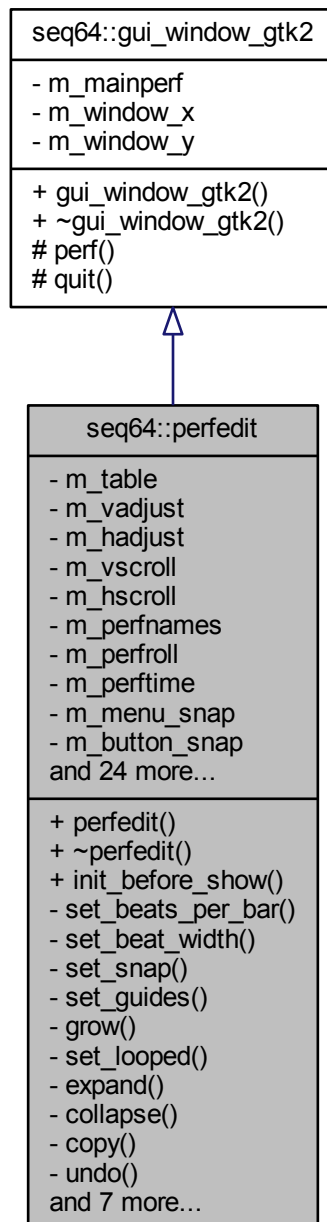
New boolean to show sequence numbers; ignored in legacy mode.

Implements [seq64::configfile](#).

9.31 seq64::perfedit Class Reference

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

Inheritance diagram for seq64::perfedit:



Public Member Functions

- [perfedit](#) ([perform](#) &p, int ppqn=SEQ64_USE_DEFAULT_PPQN)
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 perfroll function of the same name.

Private Member Functions

- void [set_beats_per_bar](#) (int bpm)
Sets the beats-per-measure text and value to the given value, and then calls [set_guides\(\)](#).
- void [set_beat_width](#) (int bw)
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 perroll 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 [perfedit::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".
- int [m_bpm](#)
The current "beats per measure" value.
- int [m_bw](#)
The current "beat width" value.
- int [m_ppqn](#)
The current "parts per quarter note" value.

- int [m_standard_bpm](#)
The standard "beats per measure" of Sequencer64, which here matches the beats-per-measure displayed in the perfrill (piano roll).
- int [m_redraw_ms](#)
Provides the timer period for the perfedit timer, used to determine the rate of redrawing.

Additional Inherited Members

9.31.1 Detailed Description

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

9.31.2 Constructor & Destructor Documentation

9.31.2.1 seq64::perfedit::perfedit (perform & p, int ppqn = SEQ64_USE_DEFAULT_PPQN)

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.

9.31.2.2 seq64::perfedit::~~perfedit ()

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

9.31.3 Member Function Documentation

9.31.3.1 void seq64::perfedit::init_before_show ()

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

9.31.3.2 void seq64::perfedit::set_beats_per_bar (int bpm) [private]

The usage of is modified was faulty. Offloaded it to the perform object to make it more foolproof. See the [perform↔::modify\(\)](#) function.

9.31.3.3 void seq64::perfedit::set_beat_width (int bw) [private]

The usage of is modified was faulty. Offloaded it to the perform object to make it more foolproof. See the [perform↔::modify\(\)](#) function.

9.31.3.4 void seq64::perfedit::set_guides () [private]

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

It's a little confusing; I assigned the label "m_standard_bpm" to the value 4 in "measure_pulse = 192 * 4 * m_bpm / m_bw", but I am not sure I understand this equation... why the extra factor of 4? That 4 appears in "c_ppqn * 4" a lot in the original code.

9.31.3.5 `void seq64::perfedit::grow () [private]`

Make sure that setting the modified flag makes sense for this operation. It doesn't seem to modify members.

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

This action opens up a space of 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 perffroll to redraw.

9.31.3.7 `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 perffroll to redraw.

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

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

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

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

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

It redraws "dirty" sequences in the perffroll and the perffnames objects, and shows draw progress on the perffroll.

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

JACK will be used if it is present and, in the application, enabled. This call also sets `rc().is_pattern_playing(true)`.

9.31.3.12 `void seq64::perfedit::stop_playing () [inline], [private]`

This call also sets `rc().is_pattern_playing(true)`.

9.31.4 Field Documentation

9.31.4.1 `int seq64::perfedit::m_bpm [private]`

Do not confuse it with BPM (beats per minute). The numerator of the time signature.

9.31.4.2 `int seq64::perfedit::m_bw [private]`

The denominator of the time signature.

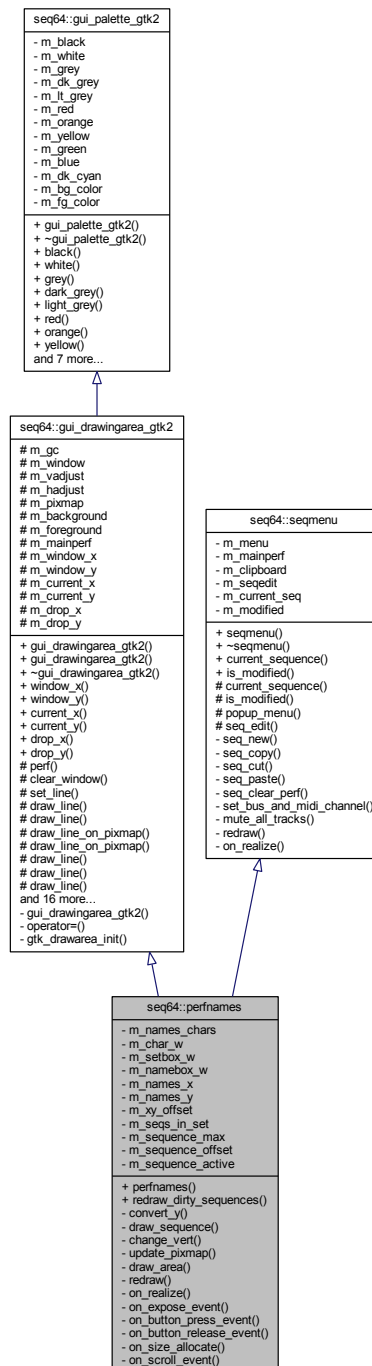
9.31.4.3 `int seq64::perfedit::m_redraw_ms [private]`

This is hardwired to 40 ms in Linux, and 20 ms in Windows.

9.32 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.

Private Attributes

- int [m_names_chars](#)
Provides the number of the characters in the name box.
- int [m_char_w](#)
Provides the "real" width of a character.
- int [m_setbox_w](#)
Provides the width of the "set number" box.
- int [m_namebox_w](#)
Provides the width of the "name" box.
- int [m_names_x](#)
Provides the width of the names box, which is the width of a character for 24 characters.
- int [m_names_y](#)
Provides the height of the names box, which is hardwired to 24 pixels.
- int [m_xy_offset](#)
Provides the horizontal and vertical offsets of the text relative to the names box.

Additional Inherited Members

9.32.1 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).

9.32.2 Constructor & Destructor Documentation

9.32.2.1 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 (now 24) in globals.h.

9.32.3 Member Function Documentation

9.32.3.1 void seq64::perfnames::draw_sequence (int seqnum) [private]

This function has to be prepared to handle an almost endless list of sequences, including unused ones, to draw them all with compatible styles. The sequences are grouped by set-number. The set-number occurs every 32 sequences in the leftmost column of the window.

9.32.3.2 void seq64::perfnames::on_realize () [private]

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

9.32.3.3 bool seq64::perfnames::on_expose_event (GdkEventExpose * ev) [private]

It draws all of the sequences.

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

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

9.32.4 Field Documentation

9.32.4.1 int seq64::perfnames::m_names_chars [private]

Pretty much hardwired to 24 at present.

9.32.4.2 int seq64::perfnames::m_char_w [private]

This value is obtained from a font-renderer accessor function.

9.32.4.3 int seq64::perfnames::m_setbox_w [private]

This used to be hardwired to 6 * 2 (character-width times two).

9.32.4.4 int seq64::perfnames::m_namebox_w [private]

This used to be a weird calculation based on character width.

9.32.4.5 int seq64::perfnames::m_names_y [private]

This value was once 22 pixels, but we need a little extra room for our new font. This extra room is compatible enough with the old font, as well.

9.32.4.6 int seq64::perfnames::m_xy_offset [private]

Currently hardwired.

9.33 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.
- bool [is_modified](#) () const
'Getter' function for member m_is_modified
- void [modify](#) ()
'Setter' function for member m_is_modified This setter only sets the modified-flag to true.
- 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](#) *pfcbb)
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 `new_sequence` (int seq)
- Creates a new pattern/sequence for the given slot, and sets the new pattern's master MIDI bus address.*
- 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, bool setstart=true)
- Set the left marker at the given tick.*
- long `get_left_tick` () const
- 'Getter' function for member `m_left_tick`*
- void `set_start_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, bool setstart=true)
- 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.*
- void `split_trigger` (int seqnum, long tick)
- Convenience function for `perroll`'s split-trigger functionality.*
- long `get_max_trigger` ()
- Locates the largest trigger value among the active sequences.*
- void `collapse` ()
- Convenience function for `perfed`'s collapse functionality.*
- void `copy` ()
- Convenience function for `perfed`'s copy functionality.*
- void `expand` ()
- Convenience function for `perfed`'s expand functionality.*

- `midi_control` & `midi_control_toggle` (int seq)
Retrieves a reference to a value from `m_midi_cc_toggle[]`.
- `midi_control` & `midi_control_on` (int seq)
Retrieves a reference to a value from `m_midi_cc_on[]`.
- `midi_control` & `midi_control_off` (int seq)
Retrieves a reference to 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_screen_set_notepad` (const std::string ¬e)
Sets the notepad text for the current screen-set.
- void `set_screenset` (int ss)
Sets the `m_screenset` value (the index or ID of the current screen set).
- int `get_screenset` () const
'Getter' function for member `m_screenset`
- 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 on all of the active sequences, and sets the group mute flag, `m_mute_group_selected`, to the clamped g-mute value.
- 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)
Makes some checks and sets the group mute flag, `m_mute_group_selected`, to the clamped g-mute value, if all goes well (no null sequences are encountered).
- 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_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.

- bool [is_active](#) (int seq)

Checks the pattern/sequence for activity.

- [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_beats_per_minute](#) (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_beats_per_minute](#) ()

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 [sequence_playing_on](#) (int seq)

Turn off the playing of a sequence, if it is active.

- void [sequence_playing_off](#) (int seq)

Turn off the playing of a sequence, if it is active.

- void [set_group_mute_state](#) (int g_track, bool mute_state)

This function sets the mute state of an element in the m_mute_group array.

- bool [get_group_mute_state](#) (int g_track)

The "inverse" of [set_group_mute_state\(\)](#), it gets the value of the desired track.

- 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()`.*
- 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_sequence_key` Provides access to `keys().show_ui_sequence_key()`.
 - bool `show_ui_sequence_number` () const
Accessor `m_show_ui_sequence_number` Provides access to `keys().show_ui_sequence_number()`.
 - bool `is_playing` () const
'Getter' function for member `rc().is_pattern_playing()` Provide a convenience function so that clients don't have to mess with a global variable when they're dealing with a perform object.
 - 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_beats_per_minute` ()
Encapsulates some calls used in `mainwnd`.
 - int `increment_beats_per_minute` ()
Encapsulates some calls used in `mainwnd`.
 - int `decrement_screenset` ()
Encapsulates some calls used in `mainwnd`.
 - int `increment_screenset` ()
Encapsulates some calls used in `mainwnd`.
 - bool `highlight` (const `sequence` &seq) const
True if a sequence is empty and should be highlighted.
 - void `sequence_key` (int seq)
Handle a sequence key to toggle the playing of an active pattern in the selected screen-set.
 - std::string `sequence_label` (const `sequence` &seq)
Provides a way to format the sequence parameters string for display in the `mainwid` or `perfnames` modules.
 - void `set_input_bus` (int bus, bool input_active)
Sets the input bus, and handles the special "key labels on sequence" and "sequence numbers on sequence" functionality.
 - bool `mainwnd_key_event` (const `keystroke` &k)
Provided for `mainwnd::on_key_press_event()` and `mainwnd::on_key_release_event()` to call.
 - bool `perfroll_key_event` (const `keystroke` &k, int drop_sequence)
Provided for `perfroll::on_key_press_event()` and `perfroll::on_key_release_event()` to call.

Private Member Functions

- bool `seq_in_playing_screen` (int seq)
A helper function for determining if the mode group is in force, the playing screenset is the same as the current screenset, and the sequence is in the range of the playing screenset.
- void `is_modified` (bool flag)
'Setter' function for member `m_is_modified` This setter is private.
- bool `is_midi_control_valid` (int seq) const

- *Checks the parameter against c_midi_controls.*
- bool [is_screenset_valid](#) (int screenset) const
Checks the screenset against m_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
- int [mute_group_offset](#) (int track)
A helper function to calculate the index into the mute-group array, based on the desired track.
- 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.
- bool [install_sequence](#) (sequence *seq, int seqnum)
A private helper function for [add_sequence\(\)](#) and [new_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.
- int [clamp_track](#) (int track) const
Provides common code to keep the track value valid.
- void [set_all_key_events](#) ()
Pass-along function for [keys\(\)](#).set_all_key_events.
- void [set_all_key_groups](#) ()
Pass-along function for [keys\(\)](#).set_all_key_events.
- 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.

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.
- int [m_playscreen_offset](#)
Playing screen sequence number offset.
- [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.
- int [m_ppqn](#)
Holds the current PPQN for usage in various actions.

- int [m_one_measure](#)
*Holds the "one measure's worth" of pulses (ticks), which is normally $m_ppqn * 4$.*
- long [m_left_tick](#)
Holds the position of the left (L) marker, and it is first defined as 0.
- long [m_right_tick](#)
Holds the position of the right (R) marker, and it is first defined as the end of the fourth measure.
- long [m_starting_tick](#)
Holds the starting tick for playing.
- long [m_tick](#)
MIDI Clock support.
- int [m_seqs_in_set](#)
We will eventually replace `c_seqs_in_set` with this member, which defaults to the value of `c_seqs_in_set`.
- int [m_max_sets](#)
A replacement for the `c_max_sets` constant.
- int [m_sequence_count](#)
Keeps track of created sequences, whether or not they are active.
- int [m_sequence_max](#)
A replacement for the `c_max_sequence` constant.
- bool [m_is_modified](#)
It may be a good idea to eventually centralize all of the dirtiness of a performance here.
- condition_var [m_condition_var](#)
A condition variable to protect...
- [jack_assistant](#) [m_jack_asst](#)
A wrapper object for the JACK support of this application.

Static Private Attributes

- static midi_control [sm_mc_dummy](#)
Provides a dummy, inactive midi_control object to handle out-of-range midi_control indicies.

Friends

- int [jack_sync_callback](#) (jack_transport_state_t state, jack_position_t *pos, void *arg)
Global functions for JACK support and JACK sessions.

9.33.1 Detailed Description

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

9.33.2 Constructor & Destructor Documentation

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

Parameters

<code>mygui</code>	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 <code>seq_gtkmm2</code> GUI-specific library. Note that we access the <code>m_gui_support</code> member using the gui() accessor function.
--------------------	---

9.33.2.2 seq64::perform::~~perform ()

Finally, any active or inactive (but allocated) patterns/sequences are deleted, and their pointer nullified..

9.33.3 Member Function Documentation

9.33.3.1 void seq64::perform::modify () [inline]

The setter that will, [is_modified\(\)](#), is private. No one but perform and its friends should falsify this flag.

9.33.3.2 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.

9.33.3.3 void seq64::perform::init ()

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

9.33.3.4 void seq64::perform::clear_all ()

The mainwnd module calls this function. Note that perform now handles the "is modified" flag on behalf of all external objects, to centralize and simplify the dirtying of a MIDI tune.

Anything else to clear? What about all the other sequence flags? We can beef up [delete_sequence\(\)](#) for them, at some point.

9.33.3.5 void seq64::perform::launch_input_thread ()

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

9.33.3.6 void seq64::perform::launch_output_thread ()

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

9.33.3.7 void seq64::perform::init_jack ()

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

9.33.3.8 void seq64::perform::new_sequence (int seq)

Then it activates the pattern [this is done in the [install_sequence\(\)](#) function]. It doesn't deal with thrown exceptions.

This function is called by the seqmenu and mainwnd objects to create a new sequence. We now pass this sequence to [install_sequence\(\)](#) to better handle potential memory leakage, and to make sure the sequence gets counted. Also, adding a new sequence from the user-interface is a significant modification, so the "is modified" flag gets set.

Parameters

<i>seq</i>	The prospective sequence number of the new sequence.
------------	--

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

No check is made for a null pointer, but the [install_sequence\(\)](#) call will make sure such a pointer is officially logged.

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 finish.

Finally, note that this function is used only by midifile, when reading in a MIDI song. Therefore, the "is modified" flag is *not* set by this function; loading a sequence from a file is not a modification that should lead to a prompt for saving the file later.

Todo Shouldn't we wrap around the sequence list if we can't find an empty sequence slot after prefnum?

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.

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

We now also solidify the deletion by setting the pointer to null after deletion, so it will blow up if accidentally accessed. The final act is to raise the "is modified" flag, since deleting an existing sequence is always a significant modification.

Now, this function obviously sets the "active" flag for the sequence to false. But there are a few other flags that are not modified; shouldn't we also falsify them here?

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

Parameters

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

9.33.3.12 void seq64::perform::set_left_tick (long tick, bool setstart = true)

We let the caller determine is this setting is a modification.

Parameters

<i>tick</i>	The tick (MIDI pulse) at which to place the left tick. If the left tick is greater than or equal to the right tick, then the right ticked is moved forward by one "measure's length" (m_ppqn * 4) past the left tick.
-------------	---

<i>setstart</i>	If true (the default, and longstanding implicit setting), then the starting tick is also set to the left tick.
-----------------	--

9.33.3.13 void seq64::perform::set_right_tick (long tick, bool setstart = true)

This setting is made only if the tick parameter is at or beyond the first measure. We let the caller determine if this setting is a modification.

Parameters

<i>tick</i>	The tick (MIDI pulse) at which to place the right tick. If less than or equal to the left tick setting, then the left tick is backed up by one "measure's worth" ($m_ppqn * 4$) worth of ticks from the new right tick.
<i>setstart</i>	If true (the default, and longstanding implicit setting), then the starting tick is also set to the left tick, if that got changed.

9.33.3.14 void seq64::perform::move_triggers (bool direction)

Parameters

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

9.33.3.15 void seq64::perform::copy_triggers ()

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

9.33.3.16 void seq64::perform::push_trigger_undo ()

Too bad we cannot yet keep track of all the undoes for the sake of properly handling the "is modified" flag.

9.33.3.17 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?

9.33.3.18 midi_control & seq64::perform::midi_control_toggle (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.
------------	---

9.33.3.19 midi_control & seq64::perform::midi_control_on (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.
------------	---

9.33.3.20 `midi_control & seq64::perform::midi_control_off (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.
------------	---

9.33.3.21 `void seq64::perform::handle_midi_control (int ctrl, bool state)`

Parameters

<i>ctrl</i>	The MIDI control value to use to perform an operation.
<i>state</i>	The state of the control, used with: <ul style="list-style-type: none"> - <code>c_midi_control_mod_replace</code> - <code>c_midi_control_mod_snapshot</code> - <code>c_midi_control_mod_queue</code> - <code>c_midi_control_mod_gmute</code> - <code>c_midi_control_mod_glearn</code>

9.33.3.22 `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.

9.33.3.23 `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.

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

It's not clear that we need to set the "is modified" flag just because we changed the screen set.

Parameters

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

9.33.3.25 void seq64::perform::set_playing_screenset ()

For each value up to m_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. Null sequences are no longer flagged as an error, they are just ignored.

Modifies m_playing_screen, and mutes the group tracks.

9.33.3.26 void seq64::perform::select_group_mute (int a_g_mute)

Null sequences are no longer flagged as an error, they are just ignored.

Parameters

<i>a_g_mute</i>	The number of the mute group, clamped to be between 0 and m_seqs_in_set-1.
-----------------	--

9.33.3.27 void seq64::perform::unset_mode_group_learn ()

Then unsets the group-learn mode flag.

9.33.3.28 void seq64::perform::select_mute_group (int a_group)

Null sequences are no longer flagged as an error, they are just ignored.

Will need to study this one more closely.

Parameters

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

9.33.3.29 void seq64::perform::start (bool state)

Parameters

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

9.33.3.30 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?

9.33.3.31 void seq64::perform::all_notes_off ()

Then flush the MIDI buss.

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

If setting it active, the [sequence::number\(\)](#) setter is called. It won't modify the sequence's internal copy of the sequence number if it has already been set.

Parameters

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

9.33.3.33 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.
------------	---

9.33.3.34 bool seq64::perform::is_dirty_main (int *seq*)

See the [sequence::is_dirty_main\(\)](#) function.

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.

9.33.3.35 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.

9.33.3.36 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.

9.33.3.37 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.

9.33.3.38 `bool seq64::perform::is_active (int seq) [inline]`

Todo We should have the sequence object keep track of its own activity and access that via a reference or pointer.

Parameters

<i>seq</i>	The pattern number. It is checked for invalidity. This can lead to "too many" (i.e. redundant) checks, but we're trying to centralize such checks in this function.
------------	---

Returns

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

9.33.3.39 `sequence* seq64::perform::get_sequence (int seq) [inline]`

Parameters

<i>seq</i>	The prospective sequence number.
------------	----------------------------------

Returns

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

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

Then flush the MIDI buss.

9.33.3.41 `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.
-------------	--

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

Parameters

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

9.33.3.43 void seq64::perform::set_beats_per_minute (int bpm)

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

It's not clear that we need to set the "is modified" flag just because we changed the beats per minute. Does this setting get saved to the MIDI file?

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.

9.33.3.44 void seq64::perform::set_sequence_control_status (int status)

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

9.33.3.45 void seq64::perform::unset_sequence_control_status (int status)

Then the given status is reversed in m_control_status.

9.33.3.46 void seq64::perform::sequence_playing_on (int seq)

Parameters

<i>seq</i>	The number of the sequence to be turned on.
------------	---

9.33.3.47 void seq64::perform::sequence_playing_off (int seq)

Parameters

<i>seq</i>	The number of the sequence to be turned off.
------------	--

9.33.3.48 void seq64::perform::set_group_mute_state (int gtrack, bool muted)

The index value is the track number offset by the number of the selected mute group (which seems equivalent to a set number) times the number of sequences in a set.

Parameters

<i>gtrack</i>	The number of the track to be muted/unmuted.
<i>muted</i>	This boolean indicates the state to which the track should be set.

9.33.3.49 bool seq64::perform::get_group_mute_state (int gtrack)

Parameters

<i>gtrack</i>	The number of the track for which the state is to be obtained. Like set_group_mute_state() , this value is offset by adding m_mute_group_selected * m_seqs_in_set.
---------------	--

Returns

Returns the value of m_mute_group[gtrack + set offset]

9.33.3.50 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.

9.33.3.51 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.
---------------	---------------------

9.33.3.52 void seq64::perform::save_playing_state ()

Inactive patterns get the value set to false.

9.33.3.53 bool seq64::perform::show_ui_sequence_key () const [inline]

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

9.33.3.54 bool seq64::perform::show_ui_sequence_number () const [inline]

Used in mainwid, optionsfile, and perform.

9.33.3.55 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.

9.33.3.56 int seq64::perform::decrement_beats_per_minute () [inline]

Actually does a lot of work in those function calls.

9.33.3.57 int seq64::perform::increment_beats_per_minute () [inline]

Actually does a lot of work in those function calls.

9.33.3.58 bool seq64::perform::highlight (const *sequence* & *seq*) const

This setting is currently a build-time option, but could be made a run-time option later.

9.33.3.59 `std::string seq64::perform::sequence_label (const sequence & seq)`

This string goes on the bottom-left of those user-interface elements.

The format of this string is something like the following example, depending on the "show sequence numbers" option. The values shown are, in this order, sequence number (if allowed), buss number, channel number, beats per bar, and beat width.

```
No sequence number:    31-16 4/4
Sequence number:      9  31-16 4/4
```

The sequence number and buss number are re 0, while the channel number is displayed re 1.

Note

Later, we could add the sequence hot-key to this string, though showing that is not much use in pernames. Also, this function is a stilted mix of direct access and access through sequence number.

Parameters

<i>seq</i>	Provides the reference to the sequence, use for getting the sequence parameters to be written to the label string.
------------	--

Returns

Returns the filled in label if the sequence is active. Otherwise, an empty string is returned.

9.33.3.60 `void seq64::perform::set_input_bus (int bus, bool active)`

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

Tricky Code See the bus parameter. We should provide two separate functions for this feature, but it is already combined into one input-callback function with a lot of other functionality in the options module.

Parameters

<i>bus</i>	If this value is greater than SEQ64_DEFAULT_BUSS_MAX (32), then it is treated as a user-interface flag (PERFORM_KEY_LABELS_ON_SEQUENCE or PERFORM_NUM_LABEL↔S_ON_SEQUENCE) that causes all the sequences to be dirtied, and thus get redrawn iwht the new user-interface setting.
<i>active</i>	Indicates whether the buss or the user-interface feature is active or inactive.

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

Returns

Returns true if the key was handled.

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

The "is modified" flag is raised if something is deleted, but we cannot yet handle the case where we undo all the changes. So, for now, we play it safe with the user, even if the user gets annoyed because he knows that he undid all the changes.

Returns

Returns true if the key was handled.

9.33.3.63 `bool seq64::perform::seq_in_playing_screen (int seq) [private]`

Parameters

<i>seq</i>	Provides the index of the desired sequence.
------------	---

Returns

Returns true if the sequence adheres to the conditions noted above.

9.33.3.64 `void seq64::perform::is_modified (bool flag) [inline],[private]`

The `modify()` setter, which is public, can only set `m_is_modified` to true.

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

Parameters

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

Returns

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

9.33.3.66 `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.

9.33.3.67 `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. Also, it is possible to have holes in the array representing inactive sequences, so that `sequencer_count()` would be too limiting.

Parameters

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

Returns

Returns true if the sequence number is valid.

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

It also evaluates the `m_seq[seq]` pointer value.

Note

Since we can have holes in the sequence array, where there are inactive sequences, we check if the sequence is even active before emitting a message about a null pointer for the sequence. We only want to see messages that indicate actual problems.

Parameters

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

Returns

Returns true if the sequence number is valid as per [is_seq_valid\(\)](#), and the sequence pointer is not null.

9.33.3.69 `bool 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. It does not set the "is modified" flag, since adding a sequence by loading a MIDI file should not set it. Compare [new_↵ sequence\(\)](#), used by mainwid and seqmenu, with [add_sequence\(\)](#), used by midifile.

Parameters

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

Returns

Returns true if a sequence was removed, or the sequence was successfully add. In other words, if a real change in sequence pointers occurred. It is up to the caller to decide if the change warrants setting the "is modified" flag.

9.33.3.70 `void seq64::perform::inner_start (bool 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.

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

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

9.33.3.72 `void seq64::perform::set_key_event (unsigned int keycode, long sequence_slot) [inline],[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.

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

It is called 32 times, corresponding the pattern/sequence slots in the Patterns window. Compare it to the [set_key_↵ _events\(\)](#) function.

9.33.4 Friends And Related Function Documentation

9.33.4.1 `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 jack_assistant object. Currently not checked for nullity, nor dynamic-casted.

9.33.5 Field Documentation

9.33.5.1 `midi_control seq64::perform::sm_mc_dummy` [static], [private]

Instantiate the dummy `midi_control` object, which is used in lieu of a null pointer.

We're taking code that basically works already, in the sense that it never seems to access a null pointer. So we're not even risking data transfers between this dummy object and the ones we really want to use.

9.33.5.2 `int seq64::perform::m_playscreen_offset` [private]

Saves some multiplications, should make the code easier to grok, and centralizes the use of `c_seqs_in_set`, which we want to be able to change at run-time, as a future enhancement.

9.33.5.3 `sequence* seq64::perform::m_seqs[c_max_sequence]` [private]

Todo First, make the sequence array a vector, and second, put allof these flags into a structure and access those members indirectly.

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

There are two, "live" and "song", but we're not yet sure what "true" indicates. It is most likely:

```
m_playback_mode == false:    live mode
m_playback_mode == true:     playback/song  mode
```

9.33.5.5 `int seq64::perform::m_one_measure` [private]

We can save some multiplications, and, more importantly, later define a more flexible definition of "one measure's worth" than simply four quarter notes.

9.33.5.6 `long seq64::perform::m_left_tick` [private]

Note that "tick" is actually "pulses".

9.33.5.7 `long seq64::perform::m_right_tick` [private]

Note that "tick" is actually "pulses".

9.33.5.8 `long seq64::perform::m_starting_tick` [private]

By default, this value is always reset to the value of the "left tick". We want to eventually be able to leave it at the last playing tick, to support a "pause" functionality. Note that "tick" is actually "pulses".

9.33.5.9 `int seq64::perform::m_seqs_in_set` `[private]`

This change will require some arrays to be dynamically allocated (vectors).

9.33.5.10 `int seq64::perform::m_max_sets` `[private]`

Again, currently set to the old value, which is used in hard-wired array sizes. To make it variable will require a move from arrays to vectors.

9.33.5.11 `int seq64::perform::m_sequence_count` `[private]`

Used by the [install_sequence\(\)](#) function.

9.33.5.12 `int seq64::perform::m_sequence_max` `[private]`

However, this value is already $32 * 32 = 1024$, and is probably enough for any usage. Famous last words?

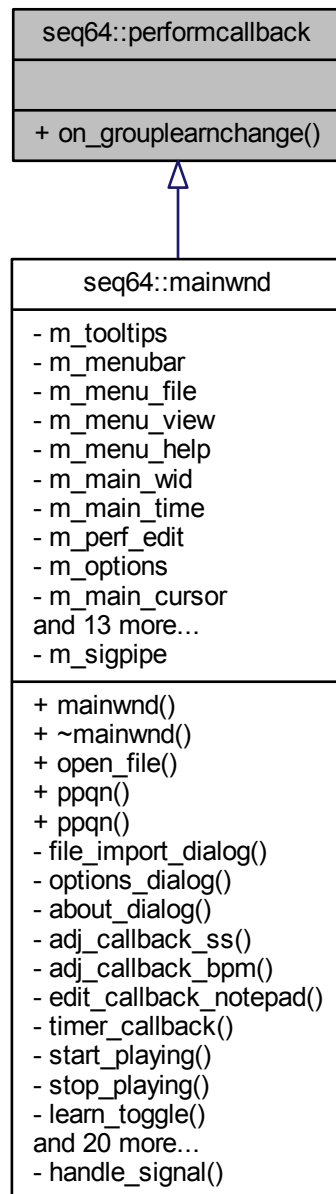
9.33.5.13 `bool seq64::perform::m_is_modified` `[private]`

All the GUIs seem to use a perform object. IN PROGRESS.

9.34 seq64::performcallback Struct Reference

Provides for notification of events.

Inheritance diagram for seq64::performcallback:



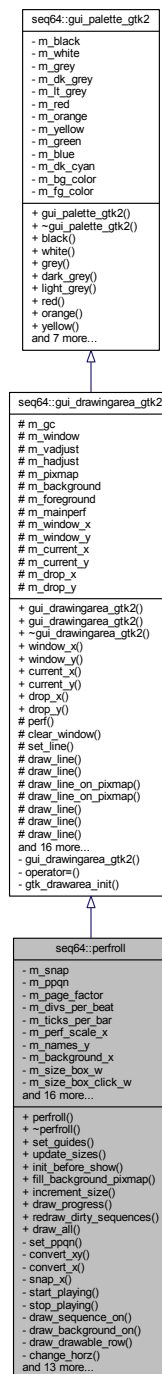
9.34.1 Detailed Description

Provide a response to a group-learn change event.

9.35 seq64::perftroll Class Reference

This class implements the performance roll user interface.

Inheritance diagram for seq64::perfroll:



Public Member Functions

- [perfroll](#) ([perform](#) &[perf](#), Gtk::Adjustment &hadjust, Gtk::Adjustment &vadjust, int ppqn=SEQ64_USE_DEF↵ AULT_PPQN)

Principal constructor.

- [~perfroll](#) ()

This destructor deletes the interaction object.

- void [set_guides](#) (int snap, int measure, int beat)
This function sets the `m_snap`, `m_measure_length`, and `m_beat_length` members directly from the function parameters, which are in units of pulses (sometimes misleadingly called "ticks".)
- 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 `perftroll_input`.

Private Member Functions

- void [set_ppqn](#) (int ppqn)
Handles changes to the PPQN value in one place.
- 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](#) (int seqnum)
Draws the given pattern/sequence on the given drawable area.
- void [draw_background_on](#) (int seqnum)
Draws the given pattern/sequence background on the given drawable area.
- void [draw_drawable_row](#) (long 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 sequence, long tick)
Splits a trigger, whatever that means.
- void [on_realize](#) ()
Provides the on-realization callback.
- bool [on_expose_event](#) (GdkEventExpose *ev)
Handles the on-expose event.
- bool [on_button_press_event](#) (GdkEventButton *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 *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 *ev)
Handles motion notification by forwarding it to the interaction object's motion-notification callback function.
- bool [on_scroll_event](#) (GdkEventScroll *ev)
Handles horizontal and vertical scrolling.
- bool [on_focus_in_event](#) (GdkEventFocus *ev)
This callback handles an in-focus event by setting the flag to HAS_FOCUS.
- bool [on_focus_out_event](#) (GdkEventFocus *ev)
This callback handles an out-of-focus event by resetting the flag HAS_FOCUS.
- void [on_size_allocate](#) (Gtk::Allocation &al)
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 *ev)
This callback function handles a key-press event.
- void [on_size_request](#) (GtkRequisition *)
This do-nothing callback effectively throws away a size request.

Private Attributes

- [FruityPerfInput m_fruity_interaction](#)
We need both styles of interaction object present.
- [Seq24PerfInput m_seq24_interaction](#)
Provides support for standard Seq24 mouse handling, plus the keystroke handlers.

Friends

- class [FruityPerfInput](#)
These friend implement interaction-specific behavior, although only the Seq24 interactions support keyboard processing, except for some common functionality provided by [perform::perfroll_key_event\(\)](#).

Additional Inherited Members

9.35.1 Constructor & Destructor Documentation

9.35.1.1 seq64::perfroll::~~perfroll ()

Well, now there are two objects, so no explicit deletion necessary.

9.35.2 Member Function Documentation

9.35.2.1 void seq64::perfroll::set_guides (int snap, int measure, int beat)

This function then fills in the background, and queues up a draw operation.

Parameters

snap	Provides the number of snap-pulses (pulses per snap interval) as calculated in perfedit::set_guides() . This is actually equal to the measure-pulses divided by the snap value in perfedit; the snap value defaults to 8.
----------------------	---

<i>measure</i>	Provides the number of measure-pulses (pulses per measure) as calculated in perfedit::set_guides() .
<i>beat</i>	Provides the number of beat-pulses (pulses per beat) as calculated in perfedit::set_guides() .

9.35.2.2 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_beat", see its definition in the class initializer list.

9.35.2.3 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.

9.35.2.4 void seq64::perfroll::fill_background_pixmap ()

This first thing done is to clear the background by painting it with a filled white rectangle.

9.35.2.5 void seq64::perfroll::set_ppqn (int ppqn) [private]

The `m_ticks_per_bar` member replaces the global `ppqn` times 16. This construct is parts-per-quarter-note times 4 quarter notes times 4 sixteenth notes in a bar. (We think...)

The `m_perf_scale_x` member starts out at `c_perf_scale_x`, which is 32 ticks per pixel at the default tick rate of 192 PPQN. We adjust this now. But note that this calculation still involves the `c_perf_scale_x` constant.

9.35.2.6 void seq64::perfroll::convert_xy (int x, int y, long & d_tick, int & d_seq) [private]

The results are returned via the `d_tick` and `d_seq` parameters.

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

The result is returned via the `tick` parameter.

9.35.2.8 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.

9.35.2.9 void seq64::perfull::start_playing () [private]

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

9.35.2.10 void seq64::perfull::stop_playing () [private]

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

9.35.2.11 void seq64::perfull::draw_sequence_on (int seqnum) [private]

Statement nesting from hell!

9.35.2.12 void seq64::perfull::draw_drawable_row (long y) [private]

It is involved in the drawing of a greyed (selected) row.

What's weird is that we divide `y` by `m_names_y`, then multiply it by `m_names_y`, before passing the result to `draw↵_drawable()`. However, if we just as `y` casted to an int, then the drawing of the row is only partial, vertically.

9.35.2.13 void seq64::perfull::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.

9.35.2.14 bool seq64::perfull::on_expose_event (GdkEventExpose * ev) [private]

Returns

Always returns true.

9.35.2.15 bool seq64::perfull::on_button_press_event (GdkEventButton * ev) [private]

This gives us Seq24 versus Fruity behavior.

One minor issue: Fruity behavior doesn't yet provide the keystroke behavior we now handle for the Seq24 mode of operation.

9.35.2.16 bool seq64::perfull::on_button_release_event (GdkEventButton * ev) [private]

This gives us Seq24 versus Fruity behavior.

9.35.2.17 bool seq64::perfull::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. Note that we pass the functionality on to the `perform::perfull_key_event()` function for the handling of delete, cut, copy, paste, and undo operations. If the keystroke is not handled by that function, then we handle it here.

Note that only the Seq24 input interaction object handles additional keystrokes not handled by the `perfull_key_↵event()` function.

9.35.3 Field Documentation

9.35.3.1 FruityPerfInput seq64::perfroll::m_fruity_interaction [private]

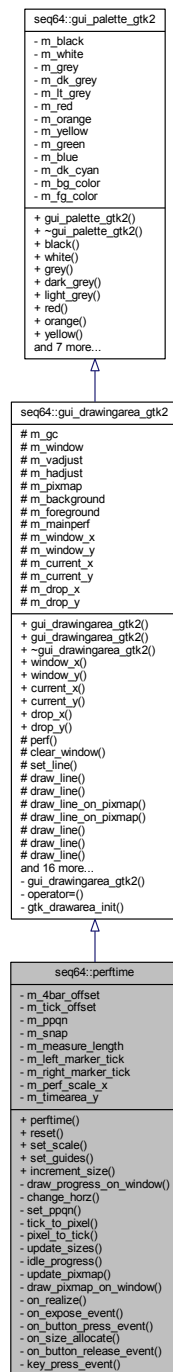
Even if the user specifies the fruity interaction, the Seq24 interaction is still needed to handle our new keystroke support for the perfroll. We need both objects to exist all the time, similar to the Fruity/Seq24 roles in the seqroll object.

Obsolete [AbstractPerfInput](#) * m_interaction

9.36 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, int ppqn=SEQ64_USE_DEFAULT_PPQN)

Principal constructor.

- void [set_guides](#) (int snap, int measure)

Sets the `m_snap` value and the `m_measure_length` members directly from the function parameters, which are in units of pulses (sometimes misleadingly called "ticks").

- void [increment_size](#) ()
This function does nothing.

Private Member Functions

- void [change_horz](#) ()
Change the `m_4bar_offset` and queue a draw operation.
- void [set_ppqn](#) (int ppqn)
Handles changes to the PPQN value in one place.
- long [tick_to_pixel](#) (long tick)
Common calculation to convert a pulse/tick value to a perftime x value.
- long [pixel_to_tick](#) (long pixel)
The inverse of [tick_to_pixel](#)().
- 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.

Private Attributes

- int [m_4bar_offset](#)
Not yet sure exactly what this member represents.
- int [m_tick_offset](#)
This member is `m_4bar_offset` times 16 times the current PPQN, to save some calculations and centralize this value.
- int [m_ppqn](#)
The current value of PPQN, which we are trying to get to work everywhere, when PPQN is changed from the global `ppqn = 192`.
- int [m_snap](#)
Snap value, starts out very small, equal to `m_ppqn`.
- int [m_measure_length](#)
Provides the length of a measure in pulses or ticks.
- int [m_left_marker_tick](#)
Holds the current location of the left (L) marker when arrow movement is in force.
- int [m_right_marker_tick](#)
Holds the current location of the right (R) marker when arrow movement is in force.
- int [m_perf_scale_x](#)
A class version of the global `c_perf_scale_x` factor.
- int [m_timearea_y](#)
A class version of the global `c_timerarea_y` factor.

Additional Inherited Members

9.36.1 Constructor & Destructor Documentation

9.36.1.1 `seq64::perftime::perftime (perform & p, Gtk::Adjustment & hadjst, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

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.

9.36.2 Member Function Documentation

9.36.2.1 `void seq64::perftime::set_guides (int snap, int measure)`

This function then fills in the background, and queues up a draw operation.

Parameters

<i>snap</i>	Provides the number of snap-pulses (pulses per snap interval) as calculated in perfedit::set_guides() . This is actually equal to the measure-pulses divided by the snap value in <code>perfedit</code> ; the snap value defaults to 8.
<i>measure</i>	Provides the number of measure-pulses (pulses per measure) as calculated in perfedit::set_guides() .

9.36.2.2 `void seq64::perftime::on_realize () [private]`

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

Done in base-class's `on_realize()` and in its constructor now.

```
m_window = get_window();
m_gc = Gdk::GC::create(m_window);
m_window->clear();
set_size_request(10, m_timearea_y);
```

9.36.2.3 `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.

9.36.3 Field Documentation

9.36.3.1 `int seq64::perftime::m_measure_length [private]`

This value is `m_ppqn * 4`, though eventually we want to employ a more flexible representation of measure length.

9.36.3.2 `int seq64::perftime::m_left_marker_tick` `[private]`

Otherwise it is -1.

9.36.3.3 `int seq64::perftime::m_right_marker_tick` `[private]`

Otherwise it is -1.

9.37 seq64::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 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.
- `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

- `std::string home_config_directory () const`

Provides the directory for the configuration file, and also creates the directory if necessary.

Private Attributes

- `std::string m_filename`

Provides the name of current MIDI file.

9.37.1 Member Function Documentation

9.37.1.1 `std::string seq64::rc_settings::home_config_directory () const` [private]

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.

9.38 `seq64::gui_drawingarea_gtk2::rect` Struct Reference

A small helper structure representing a rectangle.

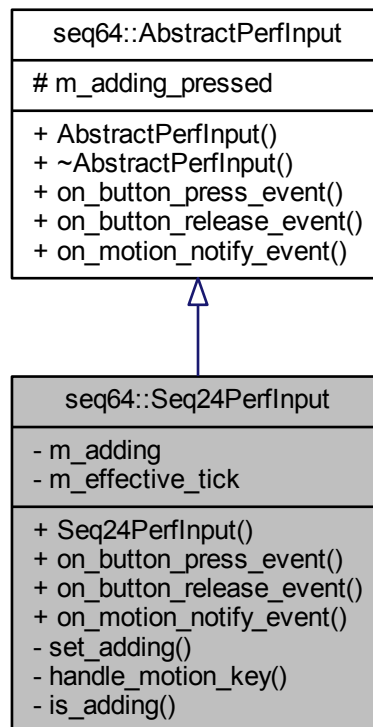
9.39 `seq64::rect` Class Reference

A small helper class representing a rectangle.

9.40 `seq64::Seq24PerfInput` Class Reference

Implements the default (Seq24) performance input characteristics of this application.

Inheritance diagram for seq64::Seq24PerfInput:



Public Member Functions

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

Private Member Functions

- void [set_adding](#) (bool a_adding, [perfroll](#) &roll)
A popup menu (which one?) calls this.
- bool [handle_motion_key](#) (bool is_left, [perfroll](#) &roll)
Handles the keystroke motion-notify event for moving a pattern back and forth in the performance.
- bool [is_adding](#) () const
'Getter' function for member m_adding

Private Attributes

- bool [m_adding](#)

Indicates we are in the middle of adding a sequence segment to the performance.

- long [m_effective_tick](#)

The current tick for the current segment?

9.40.1 Member Function Documentation

9.40.1.1 `bool seq64::Seq24PerfInput::on_button_press_event (GdkEventButton * ev, perfroll & roll)` [virtual]

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

Returns

Returns true if a modification occurred.

Implements [seq64::AbstractPerfInput](#).

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

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

Returns

Returns true if any modification occurred.

Implements [seq64::AbstractPerfInput](#).

9.40.1.3 `bool seq64::Seq24PerfInput::on_motion_notify_event (GdkEventMotion * ev, perfroll & roll)` [virtual]

Returns

Returns true if a modification occurs. This function used to always return true.

Implements [seq64::AbstractPerfInput](#).

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

What does it mean?

9.40.1.5 `bool seq64::Seq24PerfInput::handle_motion_key (bool is_left, perfroll & roll)` [private]

What happens when the mouse is used to drag the pattern is that, first, roll.m_drop_tick is set by left-clicking into the pattern to select it. As the pattern is dragged, the drop-tick value does not change, but the tick (converted from the moving x value) does.

Then the button-handler sets roll.m_moving = true, and calculates roll.m_drop_tick_trigger_offset = roll.m_drop_tick - p.get_sequence(dropseq)->selected_trigger_start();

The motion handler sees that roll.m_moving is true, gets the new tick value from the new x value, offsets it, and calls p.get_sequence(dropseq)->move_selected_triggers_to(tick, true).

When the user releases the left button, then roll.m_growing is turned of and the roll draw_all()'s.

Parameters

<i>is_left</i>	False denotes the right arrow key, and true denotes the left arrow key.
<i>roll</i>	Provides a reference to the parent roll, which keeps track of most of the information about the status of the window.

Returns

Returns true if there was some action able to happen that would necessitate a window update. We've updated [triggers::move_selected\(\)](#) [called indirectly near the end of this routine] to return false if no more movement could be made. This prevents this routine from moving way ahead after movement of the selected (in the user-interface) trigger stops.

9.41 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.

9.41.1 Member Function Documentation

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

Modifies m_adding as well.

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

Set values for dragging, then reset the box that holds dirty redraw spot. Then do the rest.

Returns

Returns true if a likely modification was made. This function used to return true all the time.

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

9.41.1.3 bool seq64::Seq24SeqEventInput::on_button_release_event (GdkEventButton * a_ev, sequevent & segev)

Returns

Returns true if a likely modification was made. This function used to return true all the time.

9.41.1.4 `bool seq64::Seq24SeqEventInput::on_motion_notify_event (GdkEventMotion * a_ev, seqevent & seqev)`

Returns

Returns true if a likely modification was made. This function used to return true all the time.

9.42 seq64::Seq24SeqRollInput Class 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)
Seq24-style on-motion mouse interaction.

9.42.1 Member Function Documentation

9.42.1.1 `void seq64::Seq24SeqRollInput::set_adding (bool adding, seqroll & scroll)`

What calls this? It is actually a right click.

9.42.1.2 `bool seq64::Seq24SeqRollInput::on_button_press_event (GdkEventButton * ev, seqroll & scroll)`

This function now uses the needs_update flag to determine if the perform object should modify().

Returns

Returns the value of needs_update. It used to return only true.

9.42.1.3 `bool seq64::Seq24SeqRollInput::on_button_release_event (GdkEventButton * ev, seqroll & scroll)`

This function now uses the needs_update flag to determine if the perform object should modify().

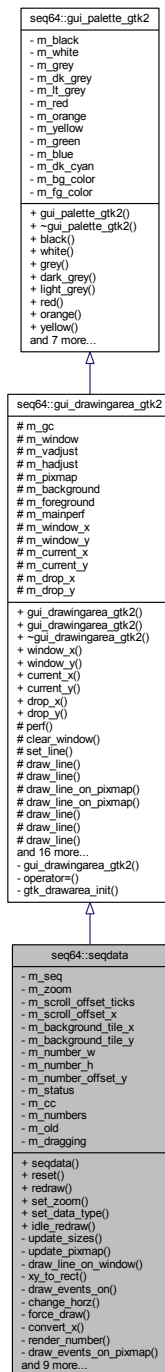
Returns

Returns the value of needs_update. It used to return only true.

9.43 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](#) (midibyte status, midibyte 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 the data window.*
- void [xy_to_rect](#) (int x1, int y1, int x2, int y2, int &rx, int &ry, int &rw, int &rh)
 - This function takes two points, and returns an Xwin rectangle, returned via the last four parameters.*
- void [draw_events_on](#) (Glib::RefPtr< Gdk::Drawable > drawable)
 - 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 [render_number](#) (Glib::RefPtr< Gdk::Pixmap > &pixmap, int x, int y, const char *const num)
 - Convenience function for rendering numbers.*
- 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 *ev)
 - Implements the on-expose event.*
- bool [on_button_press_event](#) (GdkEventButton *ev)
 - Implement a button-press event.*
- bool [on_button_release_event](#) (GdkEventButton *ev)
 - Implement a button-release event.*
- bool [on_motion_notify_event](#) (GdkEventMotion *ev)
 - Handles a motion-notify event.*
- bool [on_leave_notify_event](#) (GdkEventCrossing *ev)
 - Handles an on-leave notification event.*
- bool [on_scroll_event](#) (GdkEventScroll *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
- int [m_number_w](#)
The adjusted width of a digit in a data number.
- int [m_number_h](#)
The adjusted height of all digits in a data number.
- int [m_number_offset_y](#)
A new value to make it easier to adapt the vertical number drawing of a data item's numeric value to a different font.
- midibyte [m_status](#)
What is the data window currently editing?

Additional Inherited Members

9.43.1 Constructor & Destructor Documentation

9.43.1.1 `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.

9.43.2 Member Function Documentation

9.43.2.1 `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.

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

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

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

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

9.43.2.4 `void seq64::seqdata::set_data_type (midibyte status, midibyte control)`

Perhaps we should check that at least one of the parameters causes a change.

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

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

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

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

9.43.2.7 `void seq64::seqdata::xy_to_rect (int x1, int y1, int x2, int y2, int & rx, int & ry, int & rw, int & rh) [private]`

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

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

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

Note that this function creates a small pixmap for every possible y-value, where y ranges from 0 to MIDI_COUNT↔_MAX-1 = 127. It then fills each pixmap with a numeric representation of that y value, up to three digits (left-padded with spaces).

9.43.2.9 `bool seq64::seqdata::on_button_press_event (GdkEventButton * ev) [private]`

Returns

Always returns true.

9.43.2.10 `bool seq64::seqdata::on_button_release_event (GdkEventButton * ev) [private]`

Returns

Returns true if a modification occurred, and in that case also sets the perform modification flag.

9.43.2.11 `bool seq64::seqdata::on_motion_notify_event (GdkEventMotion * ev) [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.

Returns

Returns true if a change in event data occurred. If true, then the perform modification flag is set.

9.43.2.12 `bool seq64::seqdata::on_scroll_event (GdkEventScroll * a_ev) [private]`

This scroll event only handles basic scrolling, without any modifier keys such as SEQ64_CONTROL_MASK or SEQ64K_SHIFT_MASK.

Returns

Always returns true.

9.43.3 Field Documentation

9.43.3.1 `int seq64::seqdata::m_number_w [private]`

By "adjusted", well this is just a minor tweak for appearances.

9.43.3.2 `int seq64::seqdata::m_number_h [private]`

Basically, the character height times 3. By "adjusted", well this is just a minor tweak for appearances.

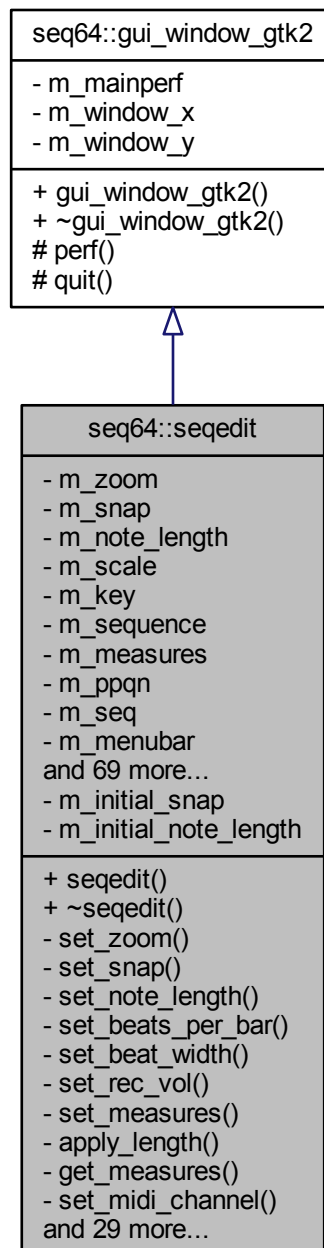
9.43.3.3 int seq64::seqdata::m_number_offset_y [private]

This value was hardwired as 8, for a character height of 10.

9.44 seq64::seqedit Class Reference

Implements the Pattern Editor, which has references to:

Inheritance diagram for seq64::seqedit:



Public Member Functions

- [seqedit](#) ([perform](#) &[perf](#), [sequence](#) &seq, int pos, int ppqn=SEQ64_USE_DEFAULT_PPQN)
Principal constructor.
- [~seqedit](#) ()
A rote destructor.

Private Member Functions

- void [set_zoom](#) (int zoom)
Selects the given zoom value.
- void [set_snap](#) (int snap)
Selects the given snap value.
- void [set_note_length](#) (int note_length)
Selects the given note-length value.
- void [set_beats_per_bar](#) (int bpm)
Set the bpm (beats per measure) value, using the given parameter, and some internal values passed to [apply_length\(\)](#).
- void [set_beat_width](#) (int bw)
Set the bw (beat width) value, using the given parameter, and some internal values passed to [apply_length\(\)](#).
- void [set_rec_vol](#) (int recvol)
Passes the given parameter to [sequence::set_rec_vol\(\)](#).
- void [set_measures](#) (int lim)
Set the measures value, using the given parameter, and some internal values passed to [apply_length\(\)](#).
- void [apply_length](#) (int bpm, int bw, int measures)
Sets the sequence 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 midichannel)
Selects the given MIDI channel parameter in the main sequence object, so that it will use that channel.
- void [set_midi_bus](#) (int midibus)
Selects the given MIDI buss parameter in the main sequence object, so that it will use that buss.
- void [set_scale](#) (int scale)
Selects the given scale value.
- void [set_key](#) (int note)
Selects the given key (signature) value.
- void [set_background_sequence](#) (int 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` (midibyte status, midibyte 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 *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 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 action, int 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 *event)
 - Handles an on-delete event.*
- bool `on_scroll_event` (GdkEventScroll *ev)
 - Handles an on-scroll event.*
- bool `on_key_press_event` (GdkEventKey *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.

- midibyte [m_editing_status](#)

Indicates what is the data window currently editing?

Static Private Attributes

- static int [m_initial_snap](#)

Static data members.

Additional Inherited Members

9.44.1 Detailed Description

- perform
- seqroll
- seqkeys
- seqdata
- seqtime
- sequevent
- sequence

This class has a metric ton of user-interface objects and other members.

9.44.2 Constructor & Destructor Documentation

9.44.2.1 `seq64::seqedit::seqedit (perform & p, sequence & seq, int pos, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

If provided, override the scale, key, and background-sequence with the values stored in the file with the sequence, if they are set to non-default values. This is a new feature.

Todo Offload most of the work into an initialization function like options does.

9.44.3 Member Function Documentation

9.44.3.1 `void seq64::seqedit::set_zoom (int z) [private]`

It is passed to the seqroll, seqtime, seqdata, and sequevent objects, as well.

The notation is in pixels:ticks, but I would prefer to use pulses/pixel (pulses per pixel). Oh well.

Finally, note that this value of zoom is saved to the "user" configuration file when Sequencer64 exit.

9.44.3.2 `void seq64::seqedit::set_snap (int snap) [private]`

It is passed to the seqroll, sequevent, and sequence objects, as well.

9.44.3.3 void seq64::seqedit::set_note_length (int *notelength*) [private]

It is passed to the seqroll object, as well.

Warning

Currently, we don't handle changes in the global PPQN after the creation of the menu. The creation of the menu hard-wires the values of note-length. To adjust for a new global PQN, we will need to store the original PPQN (`m_original_ppqn = m_ppqn`), and then adjust the notelength based on the new PPQN. For example if the new PPQN is twice as high as 192, then the notelength should double, though the text displayed in the "Note length" field should remain the same. A double value would be needed to handle the setting of a smaller `m_ppqn`. Not needed until we support a `set_ppqn()` function in this class. Another option is to rebuild the menu.

Parameters

<i>notelength</i>	Provides the note length in units of MIDI pulses. For example
-------------------	---

9.44.3.4 void seq64::seqedit::set_measures (int *lim*) [private]

Parameters

<i>lim</i>	Provides the sequence length, in measures.
------------	--

9.44.3.5 void seq64::seqedit::apply_length (int *bpm*, int *bw*, int *measures*) [private]

There's an implicit "adjust-triggers = true" parameter used in [sequence::set_length\(\)](#).

Then the seqroll, seqtime, seqdata, and seqevent objects are reset().

9.44.3.6 long seq64::seqedit::get_measures () [private]

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

9.44.3.7 void seq64::seqedit::set_midi_channel (int *midichannel*) [private]

Should this change raise the is-modified flag?

9.44.3.8 void seq64::seqedit::set_midi_bus (int *bus*) [private]

Should this change raise the is-modified flag?

9.44.3.9 void seq64::seqedit::set_scale (int *scale*) [private]

It is passed to the seqroll and seqkeys objects, as well. As a new feature, it is also passed to the sequence, so that it can be saved as part of the sequence data.

Note that the "initial value" for this parameter is a static variable that gets set to the new value, so that opening up another sequence causes the sequence to take on the new "initial value" as well. A feature, but should it be optional? Now it is, based on the setting of `usr().global_seq_feature()`.

9.44.3.10 void seq64::seqedit::set_key (int *key*) [private]

It is passed to the seqroll and seqkeys objects, as well. As a new feature, it is also passed to the sequence, so that it can be saved as part of the sequence data.

Note that the "initial value" for this parameter is a static variable that gets set to the new value, so that opening up another sequence causes the sequence to take on the new "initial value" as well. A feature, but should it be optional? Now it is, based on the setting of `usr().global_seq_feature()`.

9.44.3.11 void seq64::seqedit::set_background_sequence (int *seqnum*) [private]

As a new feature, it is also passed to the sequence, so that it can be saved as part of the sequence data, but only if less or equal to the maximum single-byte MIDI value, 127.

Note that the "initial value" for this parameter is a static variable that gets set to the new value, so that opening up another sequence causes the sequence to take on the new "initial value" as well. A feature, but should it be optional? Now it is, based on the setting of `usr().global_seq_feature()`.

Todo Make the sequence pointer a reference.

9.44.3.12 void seq64::seqedit::name_change_callback () [private]

That name is the name the user has given to the sequence being edited.

9.44.3.13 void seq64::seqedit::set_data_type (midibyte *status*, midibyte *control* = 0) [private]

This function uses the hardwired array `c_controller_names`.

Parameters

<i>status</i>	The current editing status.
<i>control</i>	The control value. However, we really need to validate it!

9.44.3.14 void seq64::seqedit::create_menus () [private]

The first menu is the Zoom menu, represented in the pattern/sequence editor by a button with a magnifying glass. The values are "pixels to ticks", where "ticks" are actually the "pulses" of "pulses per quarter note". We would prefer the notation "n" instead of "1:n", as in "n pulses per pixel". The Snap menu is actually the Grid Snap button, which shows two arrows pointing to a central bar.

The note-length menu is on the button that shows four notes.

This menu lets one set the key of the sequence, and is brought up by the button with the "golden key" image on it.

This button shows a down around for the bottom half of the time signature. It's tooltip is "Time signature. Length of beat." But it is called `bw`, or beat width, in the code.

This menu is shown when pressing the button at the bottom of the window that has "Vol" as its label. Let's show the numbers as well to help the user. And we'll have to document this change.

This menu sets the scale to show on the panel, and the button shows a "staircase" image. See the `c_music_scales` enumeration defined in the `globals` module.

This section sets up two different menus. The first is `m_menu_length`. This menu lets on set the sequence length in bars (not the MIDI channel). The second menu is the `m_menu_bpm`, or BPM, which here means "beats per measure" (not "beats per minute").

9.44.3.15 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.

9.44.3.16 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.

9.44.3.17 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 for screen-set 0. If more screensets have active sequences, then their screen-set number appears in the screen-set section of the menu.

Now, at present, we can only save background sequence numbers that are less than 128, which means the sequences from 0 to 127, or the first four screen sets. Higher sequences can be selected, but, right now, they cannot be saved. We'll probably fix that at some point, low priority.

9.44.3.18 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.

9.44.3.19 void seq64::seqedit::do_action (int *action*, int *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).

9.44.3.20 bool seq64::seqedit::on_delete_event (GdkEventAny * *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.

9.44.4 Field Documentation

9.44.4.1 int seq64::seqedit::m_initial_snap [static], [private]

These items apply to all of the instances of `seqedit`, and are passed on to the following constructors:

- `seqdata`
- `seqevent`
- `seqroll`

- seqtime

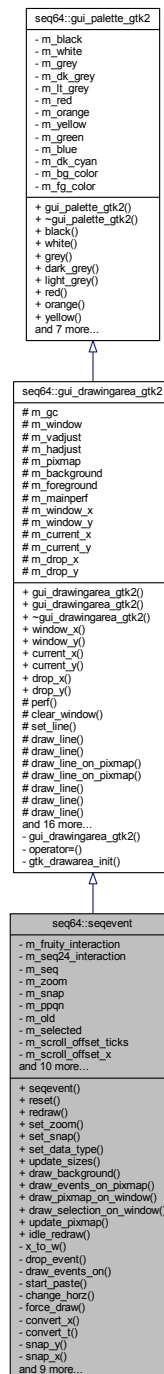
The snap and note-length defaults would be good to write to the "user" configuration file. The scale and key would be nice to write to the proprietary section of the MIDI song. Or, even more flexibly, to each sequence, if that makes sense to do, since all tracks would generally be in the same key. Right, Charles Ives?

Note that, currently, that some of these "initial values" are modified, so that they are "contagious". That is, the next sequence to be opened in the sequence editor will adopt these values. This is a long-standing feature of Seq24, but strikes us as a bit too surprising and tricky.

9.45 seq64::sequevent Class Reference

Implements the piano event drawing area.

Inheritance diagram for seq64::sequevent:



Public Member Functions

- `seqevent` (`perform` &p, `sequence` &seq, 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` (midibyte a_status, midibyte 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, the 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 corrdinate, 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.
- midibyte [m_status](#)
Indicates what is the data window currently editing?

Additional Inherited Members

9.45.1 Member Function Documentation

9.45.1.1 void seq64::sequest::set_snap (int a_snap) [inline]

Simply sets the snap member.

9.45.1.2 void seq64::sequest::set_data_type (midibyte status, midibyte control = 0)

Then redraws.

9.45.1.3 void seq64::sequest::update_sizes ()

This ends up filling the background with dotted lines, etc.

9.45.1.4 void seq64::sequest::draw_background ()

It sets the foreground to white, draws the rectangle, in order to clear the pixmap.

9.45.1.5 void seq64::sequest::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.
```

9.45.1.6 `int seq64::segevent::idle_redraw ()`

Who calls this routine?

9.45.1.7 `void seq64::segevent::x_to_w (int a_x1, int a_x2, int & a_x, int & a_w) [private]`

Then it fills in x and the width.

9.45.1.8 `void seq64::segevent::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`.

9.45.1.9 `void seq64::segevent::draw_events_on (Glib::RefPtr<Gdk::Drawable> draw) [private]`

Parameters

<i>draw</i>	The given drawable object.
-------------	----------------------------

9.45.1.10 `void seq64::segevent::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.

9.45.1.11 `void seq64::segevent::convert_x (int x, long & tick) [inline],[private]`

Why not just return it normally?

9.45.1.12 `void seq64::segevent::convert_t (long ticks, int & x) [inline],[private]`

Why not just return it normally?

9.45.1.13 `void seq64::segevent::snap_x (int & x) [private]`

- `snap` = number pulses to snap to
- `m_zoom` = number of pulses per pixel
- Therefore `snap / m_zoom` = number of pixels to snap to.

9.45.1.14 `void seq64::segevent::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.

9.45.1.15 `bool seq64::segevent::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. Added some fixes, but then commented them out until better testing can be done.

9.45.1.16 `bool seq64::seqevent::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. Added some fixes, but then commented them out until better testing can be done.

9.45.1.17 `bool seq64::seqevent::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. Added some fixes, but then commented them out until better testing can be done.

9.45.1.18 `bool seq64::seqevent::on_key_press_event (GdkEventKey * ev) [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. :-)

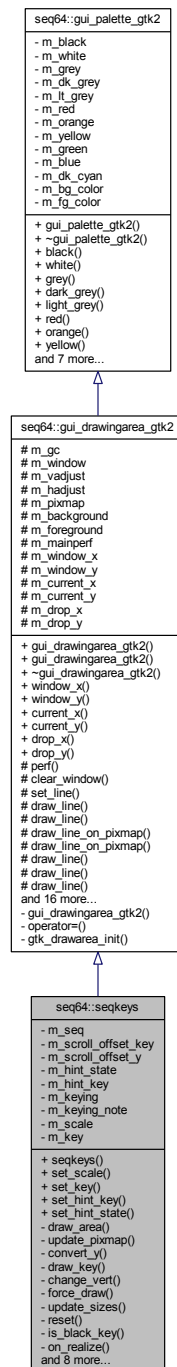
Returns

Returns true if an event was handled. Only some of the handled events also cause the perform modification flag to be set as a side-effect.

9.46 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.

- bool [is_black_key](#) (int key) const

Detects a black key.

- 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

9.46.1 Member Function Documentation

9.46.1.1 void seq64::seqkeys::set_hint_state (bool *state*)

Parameters

<i>state</i>	Provides the value for hinting, where true == on, false == off.
--------------	---

9.46.1.2 void seq64::seqkeys::draw_key (int key, bool state) [private]

It accounts for the black keys and the white keys, and for the highlighting of the active key.

Parameters

<i>key</i>	The key to be drawn.
<i>state</i>	How the key is to be drawn, where false == normal, true == grayed. A key is grayed when the mouse cursor is at the same vertical location on the piano as the key.

9.46.1.3 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.

9.46.1.4 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.

Returns

Always returns true.

9.46.1.5 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

Returns

Always returns true. piano keyboard, to make it play, and turns off the playing of the note.

Always returns true.

9.46.1.6 bool seq64::seqkeys::on_motion_notify_event (GdkEventMotion * a_p0) [private]

This allows rolling down the keyboard, playing the notes one-by-one.

Returns

Always returns false.

9.46.1.7 bool seq64::seqkeys::on_enter_notify_event (GdkEventCrossing * p0) [private]

I think this greys the current key.

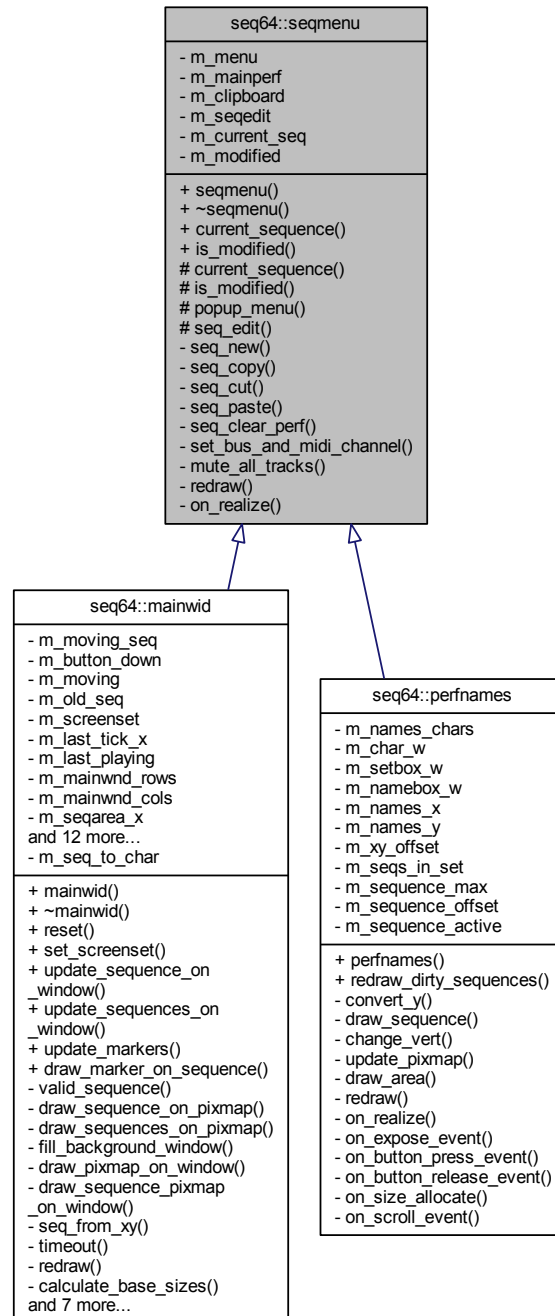
9.46.1.8 bool seq64::seqkeys::on_leave_notify_event (GdkEventCrossing * p0) [private]

I think this un-greys the current key.

9.47 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.
- void `seq_edit ()`
This menu callback launches the sequence-editor (pattern editor) window.

Private Member Functions

- 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.

9.47.1 Detailed Description

It is an abstract base class.

9.47.2 Constructor & Destructor Documentation

9.47.2.1 seq64::seqmenu::seqmenu (perform & 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.

9.47.2.2 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.

9.47.3 Member Function Documentation

9.47.3.1 void seq64::seqmenu::popup_menu () [protected]

It also sets up the pattern popup menu entries that are used in mainwid.

9.47.3.2 void seq64::seqmenu::seq_edit () [protected]

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!

9.47.3.3 void seq64::seqmenu::seq_copy () [private]

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

9.47.3.4 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.

9.47.3.5 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.

9.47.3.6 void seq64::seqmenu::seq_clear_perf () [private]

Todo All of seq_paste() can be offloaded to a (new) perform member function.

9.47.4 Field Documentation

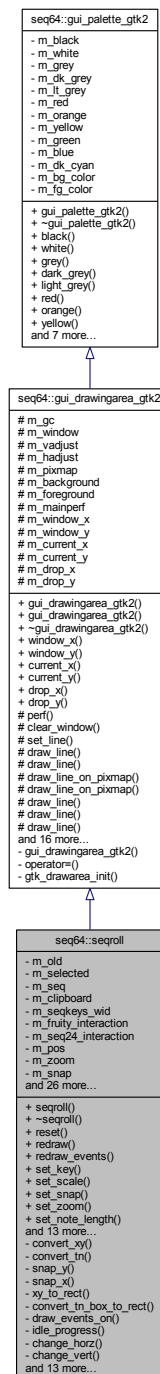
9.47.4.1 `seqedit*` `seq64::seqmenu::m_seqedit` `[private]`

We'll save the result of that function here, and will let valgrind tell us later if Gtkmm takes care of it.

9.48 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](#) (midibyte status, midibyte 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 selecton 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.
- void [update_and_draw](#) (int force=false)
Wraps up some common code.

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

- [FruitySeqRollInput m_fruity_interaction](#)
Provides a fruity input object, whether it is needed or not.
- [Seq24SeqRollInput m_seq24_interaction](#)
Provides a normal seq24 input object, which is always needed to handle, for example, keystroke input.
- int [m_zoom](#)
*one pixel == m_zoom ticks**

- midibyte [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.

Friends

- class [FruitySeqRollInput](#)
These friend implement interaction-specific behavior, although only the Seq24 interactions support keyboard processing.

Additional Inherited Members

9.48.1 Member Function Documentation

9.48.1.1 void seq64::seqroll::reset ()

It's almost identical to the [change_horz\(\)](#) function!

9.48.1.2 void seq64::seqroll::redraw_events ()

Almost: `update_and_draw(true)` are almost replaceable by `update_background()`; `update_pixmap()`; `force_draw()`;

9.48.1.3 void seq64::seqroll::set_data_type (midibyte *status*, midibyte *control*)

Unlike the same function in `sequevent`, this version does not redraw.

9.48.1.4 void seq64::seqroll::set_background_sequence (bool *state*, int *seq*)

The `state` parameter sets the boolean `m_drawing_background_seq`.

Parameters

<i>state</i>	If true, the background sequence will be drawn.
--------------	---

`seq` Provides the sequence number, which is checked against the `SEQ64_IS_LEGAL_SEQUENCE()` macro before being used. This macro allows the value `SEQ64_SEQUENCE_LIMIT`, which disables the background sequence.

9.48.1.5 void seq64::seqroll::update_sizes ()

Old comments:

```
Use m_zoom and i % m_seq->get_bpm() == 0,
int numberLines = 128 / m_seq->get_bw() / m_zoom;
int distance = c_ppqn / 32;
```

9.48.1.6 void seq64::seqroll::update_background ()

The first thing done is to clear the background, painting it white.

9.48.1.7 `void seq64::seqroll::draw_events_on_pixmap ()`

Just calls [draw_events_on\(\)](#).

9.48.1.8 `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()`.

9.48.1.9 `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.

9.48.1.10 `void seq64::seqroll::draw_events_on (Glib::RefPtr< Gdk::Drawable > draw) [private]`

"Method 0" seems be the one that draws the background sequence, if active. "Method 1" draws the sequence itself.

9.48.1.11 `bool seq64::seqroll::on_key_press_event (GdkEventKey * ev) [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).

Concerning the usage of the arrow keys in this function: This code is reached, but has no visible effect. Why? I think they were meant to move the point for playback. We may HAVE A BUG with our new handling of triggers, or maybe these depend upon the proper playback mode. In any case, the old functionality is preserved. However, if there are notes selected, then these keys support selection movement.

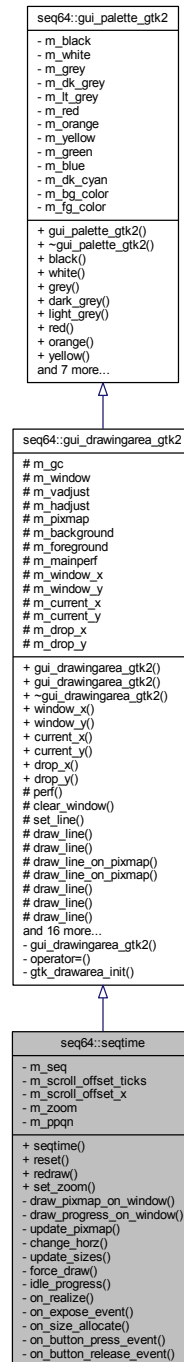
9.48.1.12 `bool seq64::seqroll::on_scroll_event (GdkEventScroll * ev) [private]`

This scroll event only handles basic scrolling without any modifier keys such as `SEQ64_CONTROL_MASK` or `SEQ64_SHIFT_MASK`.

9.49 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

9.49.1 Member Function Documentation

9.49.1.1 bool `seq64::seqtime::on_button_press_event` (GdkEventButton *) [inline],[private]

Simply returns false.

9.49.1.2 bool `seq64::seqtime::on_button_release_event` (GdkEventButton *) [inline],[private]

Simply returns false.

9.50 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` }

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*
- bool [any_selected_notes](#) () const
 - 'Getter' function for member m_events.any_selected_notes()*
- [triggers::List](#) & [triggerlist](#) ()
 - 'Getter' function for member m_triggers*
- int [number](#) () const
 - 'Getter' function for member m_seq_number*
- void [number](#) (int seqnum)
 - 'Setter' function for member m_seq_number This setter will set the sequence number only if it has not already been set.*
- 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](#) ()
 - Calls [triggers::push_undo\(\)](#) with locking.*
- void [pop_trigger_undo](#) ()
 - Calls [triggers::pop_undo\(\)](#) with locking.*
- 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_beats_per_bar](#) (long beatspermeasure)
 - 'Setter' function for member m_time_beats_per_measure*
- long [get_beats_per_bar](#) () const
 - 'Getter' function for member m_time_beats_per_measure*
- void [set_beat_width](#) (long beatwidth)
 - 'Setter' function for member m_time_beat_width*
- long [get_beat_width](#) () 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*
- bool `check_queued_tick` (long tick) const
 - Helper function for perform.*
- 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.*

- midibyte [get_midi_channel](#) () const
'Getter' function for member m_midi_channel
- void [set_midi_channel](#) (midibyte 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 len, 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 len)
Grows a trigger.
- void [del_trigger](#) (long tick)
Deletes a trigger, that brackets the given tick, from the trigger-list.
- bool [get_trigger_state](#) (long tick)
Checks the list of triggers against the given tick.
- bool [select_trigger](#) (long tick)
Checks the list of triggers against the given tick.
- bool [unselect_triggers](#) ()
Unselects all triggers.
- bool [intersect_triggers](#) (long position, long &start, long &end)
This function examines each trigger in the trigger list.
- bool [intersect_notes](#) (long position, long position_note, long &start, long &end, long ¬e)
This function examines each note in the event list.
- bool [intersect_events](#) (long posstart, long posend, long status, long &start)
This function examines each non-note event in the event list.
- void [del_selected_trigger](#) ()
Deletes the first selected trigger that is found.
- void [cut_selected_trigger](#) ()
Copies and deletes the first selected trigger that is found.
- void [copy_selected_trigger](#) ()
Copies the first selected trigger that is found.
- void [paste_trigger](#) ()
If there is a copied trigger, then this function grabs it from the trigger clipboard and adds it.
- bool [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 last-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)
- Copies triggers to...*

 - 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, midibyte status, midibyte cc, [select_action_e](#) action)

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

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

Counts the selected notes in the event list.
- int [get_num_selected_events](#) (midibyte status, midibyte cc) const

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 deltatick, int deltanote)

Removes and adds reads selected in position.
- void [add_note](#) (long tick, long len, 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, midibyte status, midibyte d0, midibyte 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.
- bool [change_event_data_range](#) (long tick_s, long tick_f, midibyte status, midibyte cc, int d_s, int d_f)

Changes the event data range.
- void [increment_selected](#) (midibyte status, midibyte control)

Increments events the match the given status and control values.
- void [decrement_selected](#) (midibyte status, midibyte control)

Decrements events the match the given status and control values.
- void [grow_selected](#) (long deltatick)

Moves note off event.
- void [stretch_selected](#) (long deltatick)

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` ()
Sets the draw-trigger iterator to the beginning of the trigger list.
- 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` ()
Goes through the list of notes, and picks the one with the lowest value.
- int `get_highest_note_event` ()
Goes through the list of notes, and picks the one with the highest value.
- bool `get_next_event` (midibyte status, midibyte cc, long *tick, midibyte *d0, midibyte *d1, bool *selected)
Get the next event in the event list that matches the given status and control character.
- bool `get_next_event` (midibyte *status, midibyte *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 `quantize_events` (midibyte status, midibyte cc, long snap_tick, int divide, bool linked=false)
Not deleting the ends, not selected.
- void `transpose_notes` (int steps, int scale)
Transposes notes by the given steps, in accordance with the given scale.
- midibyte `musical_key` () const
'Getter' function for member m_musical_key
- void `musical_key` (int key)
'Setter' function for member m_musical_key
- midibyte `musical_scale` () const
'Getter' function for member m_musical_scale
- void `musical_scale` (int scale)
'Setter' function for member m_musical_scale
- long `background_sequence` () const

'Getter' function for member `m_background_sequence`

- void `background_sequence` (int bs)

'Setter' function for member `m_background_sequence` Only partial validation at present, we do not want the upper limit to be hard-wired at this time.

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 `remove_all` ()

Clears all events from the event container.

- void `set_trigger_offset` (long trigger_offset)

Sets `m_trigger_offset` and wraps it to `m_length`.

- void `split_trigger` (trigger &trig, long splittick)

Splits the trigger given by the parameter into two triggers.

- void `adjust_trigger_offsets_to_length` (long newlen)

Adjusts trigger offsets to the length of ???, for all triggers, and undo triggers.

- 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.

- midibyte `m_midi_channel`

Contains the proper MIDI channel for this sequence.

- midibyte `m_bus`

Contains the proper MIDI bus number for this sequence.

- bool `m_song_mute`

Provides a flag for the song playback mode muting.

- int `m_notes_on`

Provides a member to hold the polyphonic step-edit note counter.

- mastermidibus * `m_masterbus`

Provides the master MIDI buss which handles the output of the sequence to the proper buss and MIDI channel.

- int `m_playing_notes` [SEQ64_MIDI_NOTES_MAX]

Provides a "map" for Note On events.

- bool `m_was_playing`

Indicates if the sequence was playing.

- bool `m_playing`

True if sequence playback currently is in progress for this sequence.

- bool `m_recording`

True if sequence recording currently is in progress for this sequence.

- bool `m_dirty_main`

These flags indicate that the content of the sequence has changed due to recording, editing, performance management, or even (?) a name change.

- bool `m_editing`

Indicates that the sequence is currently being edited.

- std::string `m_name`

Provides the name/title for the sequence.

- long [m_last_tick](#)
These members manage where we are in the playing of this sequence, including triggering.
- const int [m_maxbeats](#)
This constant provides ...?
- int [m_ppqn](#)
Holds the PPQN value for this sequence, so that we don't have to rely on a global constant value.
- int [m_seq_number](#)
A new member so that the sequence number is carried along with the sequence.
- long [m_length](#)
Holds the length of the sequence in pulse.
- long [m_time_beats_per_measure](#)
These members are used by the sequence editor to mark things in correct time on the user-interface.
- midibyte [m_musical_key](#)
Holds a copy of the musical key for this sequence, which we now support writing to this sequence.
- midibyte [m_musical_scale](#)
Holds a copy of the musical scale for this sequence, which we now support writing to this sequence.
- long [m_background_sequence](#)
Holds a copy of the background sequence number for this sequence, which we now support writing to this sequence.
- 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.

9.50.1 Detailed Description

More members than you can shake a stick at.

9.50.2 Member Enumeration Documentation

9.50.2.1 `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.

9.50.3 Member Function Documentation

9.50.3.1 `sequence & seq64::sequence::operator= (const sequence & rhs)`

Follows the stock rules for such an operator, but does a little more than 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

9.50.3.2 `int seq64::sequence::event_count () const`

Threadsafe

9.50.3.3 `void seq64::sequence::push_undo ()`

Threadsafe

9.50.3.4 `void seq64::sequence::pop_undo ()`

Threadsafe

9.50.3.5 `void seq64::sequence::pop_redo ()`

Threadsafe

9.50.3.6 `void seq64::sequence::push_trigger_undo ()`

Threadsafe

9.50.3.7 `void seq64::sequence::set_beats_per_bar (long beatspermeasure)`

Threadsafe

Parameters

<i>beatspermeasure</i>	The new setting of the beats-per-bar value.
------------------------	---

9.50.3.8 `void seq64::sequence::set_beat_width (long beatwidth)`

Threadsafe

Parameters

<i>beatwidth</i>	The new setting of the beat width value.
------------------	--

9.50.3.9 `long seq64::sequence::get_beat_width () const` `[inline]`

Threadsafe

9.50.3.10 `void seq64::sequence::set_rec_vol (long recvol)`

Threadsafe

Parameters

<i>recvol</i>	The new setting of the recording volume setting.
---------------	--

9.50.3.11 `const char* seq64::sequence::get_name () const` `[inline]`

Deprecated

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

This function is called in [seqedit::apply_length\(\)](#), when the user selects a sequence length in measures. That function calculates the length in ticks:

```

L = M x B x 4 x P / W
L == length (ticks or pulses)
M == number of measures
B == beats per measure
P == pulses per quarter-note
W == beat width in beats per measure

```

For our "b4uacuse" MIDI file, M can be about 100 measures, B is 4, P can be 192 (but we want to support higher values), and W is 4. So $L = 100 * 4 * 4 * 192 / 4 = 76800$ ticks. Seems small.

Threadsafe

9.50.3.13 `void seq64::sequence::set_playing (bool p)`

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

Parameters

<i>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.
----------	--

9.50.3.14 `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

9.50.3.15 `void seq64::sequence::off_queued ()`

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

Threadsafe

9.50.3.16 `void seq64::sequence::set_recording (bool r)`

Threadsafe

9.50.3.17 `void seq64::sequence::set_snap_tick (int st)`

Threadsafe

9.50.3.18 void seq64::sequence::set_quantized_rec (bool *qr*)

Threadsafe

9.50.3.19 void seq64::sequence::set_thru (bool *r*)

Threadsafe

9.50.3.20 bool seq64::sequence::is_dirty_main ()

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

Threadsafe

9.50.3.21 bool seq64::sequence::is_dirty_edit ()

Threadsafe

9.50.3.22 bool seq64::sequence::is_dirty_perf ()

Threadsafe

9.50.3.23 bool seq64::sequence::is_dirty_names ()

Threadsafe

9.50.3.24 void seq64::sequence::set_dirty_mp ()

Not threadsafe

9.50.3.25 void seq64::sequence::set_dirty ()

Threadsafe

9.50.3.26 void seq64::sequence::set_midi_channel (midibyte *ch*)

Threadsafe

9.50.3.27 void seq64::sequence::print ()

Not threadsafe

9.50.3.28 void seq64::sequence::print_triggers ()

Not threadsafe

9.50.3.29 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.

Parameters

<i>tick</i>	Provides the current end-tick value.
<i>playback_mode</i>	Provides how playback is managed. We think it goes like this: True indicates that it is live playback, controlled by the main windows and its layout of patterns and triggers. False indicate that the performance/song editor is in control of playback.

Threadsafe

9.50.3.30 void seq64::sequence::set_orig_tick (long tick)

Threadsafe

9.50.3.31 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. Well, actually, certain meta-events are also read, to obtain channel, buss, and more settings. Also read for a sequence, if the global-sequence flag is not set, are the new key, scale, and background sequence parameters.

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/usable.

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. Actually, this is true only in Seq24, we've fixed that behavior for Sequencer64.

9.50.3.32 void seq64::sequence::add_trigger (long tick, long len, long offset = 0, bool fixoffset = true)

A pass-through function that calls [triggers::add\(\)](#).

9.50.3.33 void seq64::sequence::split_trigger (long splittick)

This is the public overload of split_trigger.

Threadsafe

9.50.3.34 void seq64::sequence::grow_trigger (long tickfrom, long tickto, long len)

Parameters

<i>tickfrom</i>	The desired from-value back which to expand the trigger, if necessary.
<i>tickto</i>	The desired to-value towards which to expand the trigger, if necessary.
<i>len</i>	The additional length to append to tickto for the check.

Threadsafe

9.50.3.35 void seq64::sequence::del_trigger (long tick)

Threadsafe

9.50.3.36 `bool seq64::sequence::get_trigger_state (long tick)`

If any trigger is found to bracket that tick, then true is returned.

Parameters

<i>tick</i>	Provides the tick of interest.
-------------	--------------------------------

Returns

Returns true if a trigger is found that brackets the given tick.

9.50.3.37 bool seq64::sequence::select_trigger (long *tick*)

If any trigger is found to bracket that tick, then true is returned, and the trigger is marked as selected.

Parameters

<i>tick</i>	Provides the tick of interest.
-------------	--------------------------------

Returns

Returns true if a trigger is found that brackets the given tick.

9.50.3.38 bool seq64::sequence::unselect_triggers ()

Returns

Always returns false.

9.50.3.39 bool seq64::sequence::intersect_triggers (long *position*, long & *start*, long & *ender*)

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 of the matching trigger.
<i>ender</i>	The destination for the ending tick 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.

9.50.3.40 bool seq64::sequence::intersect_notes (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 timestamp of the matching note.
<i>ender</i>	The destination for the ending timestamp of the matching note.
<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.

9.50.3.41 bool seq64::sequence::intersect_events (long *posstart*, long *posend*, long *status*, long & *start*)

If the given position is between the current notes's timestamp-start and timestamp-end values, the these values are copied to the posstart and posend 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 timestamp of the matching trigger.

Returns

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

9.50.3.42 void seq64::sequence::paste_trigger ()

Why isn't this protected by a mutex? We will eventually enable this see if anything bad happens, such as a deadlock, or corruption.

9.50.3.43 bool seq64::sequence::move_selected_triggers_to (long *tick*, bool *adjustoffset*, int *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

Returns

Returns the value of [triggers::move_selected\(\)](#), which indicate that the movement could be made. Used in [Seq24PerfInput::handle_motion_key\(\)](#).

9.50.3.44 long seq64::sequence::selected_trigger_start ()

Threadsafe

Returns

Returns the tick_start() value of the last-selected trigger. If no triggers are selected, then -1 is returned.

9.50.3.45 `long seq64::sequence::selected_trigger_end ()`

Threadsafe

9.50.3.46 `long seq64::sequence::get_max_trigger ()`

Threadsafe

9.50.3.47 `void seq64::sequence::move_triggers (long starttick, long distance, bool direction)`

Note the dependence on the `m_length` member being kept in sync with the parent's value of `m_length`.

Threadsafe

9.50.3.48 `void seq64::sequence::copy_triggers (long starttick, long distance)`

Threadsafe

9.50.3.49 `void seq64::sequence::clear_triggers ()`

Threadsafe

9.50.3.50 `void seq64::sequence::set_midi_bus (char mb)`

Threadsafe

9.50.3.51 `void seq64::sequence::set_master_midi_bus (mastermidibus * mmb)`

Threadsafe

Parameters

<i>mmb</i>	Provides a pointer to the master MIDI buss for this sequence. This should be a reference.
------------	---

9.50.3.52 `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

9.50.3.53 `int seq64::sequence::select_events (long tick_s, long tick_f, midibyte status, midibyte cc, select_action_e action)`

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

Threadsafe

9.50.3.54 `int seq64::sequence::select_events (midibyte status, midibyte 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.

9.50.3.55 `int seq64::sequence::get_num_selected_notes () const`

Threadsafe

9.50.3.56 `int seq64::sequence::get_num_selected_events (midibyte status, midibyte cc) const`

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

Threadsafe

9.50.3.57 `void seq64::sequence::select_all ()`

Threadsafe

9.50.3.58 `void seq64::sequence::copy_selected ()`

Threadsafe

9.50.3.59 `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

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

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

The `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

9.50.3.61 `void seq64::sequence::add_event (long tick, midibyte status, midibyte d0, midibyte d1, bool 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

9.50.3.62 `void seq64::sequence::stream_event (event * ev)`

Threadsafe

9.50.3.63 `bool seq64::sequence::change_event_data_range (long tick_s, long tick_f, midibyte status, midibyte 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.

Parameters

<i>tick_s</i>	Provides the starting tick value.
<i>tick_f</i>	Provides the ending tick value.
<i>status</i>	Provides the event status that is to be changed.
<i>cc</i>	Provides the event control value.
<i>data_s</i>	Provides the starting data value.
<i>data_f</i>	Provides the finishing data value.

Returns

Returns true if the data was changed.

9.50.3.64 void seq64::sequence::increment_selected (midibyte *astat*, midibyte *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***9.50.3.65 void seq64::sequence::decrement_selected (midibyte *astat*, midibyte *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

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

Threadsafe

9.50.3.67 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

9.50.3.68 void seq64::sequence::remove_marked ()

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

Threadsafe

9.50.3.69 void seq64::sequence::mark_selected ()

Threadsafe

9.50.3.70 void seq64::sequence::unpaint_all ()

Threadsafe

9.50.3.71 void seq64::sequence::unselect ()

Threadsafe

9.50.3.72 void seq64::sequence::verify_and_link ()

Threadsafe

9.50.3.73 void seq64::sequence::link_new ()

Threadsafe

9.50.3.74 void seq64::sequence::zero_markers ()

This function is used when the sequencer stops.

Threadsafe

9.50.3.75 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

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

Threadsafe

9.50.3.77 void seq64::sequence::off_playing_notes ()

Threadsafe

9.50.3.78 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

9.50.3.79 void seq64::sequence::reset_draw_trigger_marker ()

Threadsafe

9.50.3.80 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.

9.50.3.81 int seq64::sequence::get_lowest_note_event ()

Threadsafe

Returns

Returns the note with the lowest value. If there are no notes in the list, then SEQ64_MIDI_COUNT_MAX-1 is returned, which of course doesn't tell the caller much.

9.50.3.82 int seq64::sequence::get_highest_note_event ()

Threadsafe

Returns

Returns the note with the highest value. If there are no notes in the list, then 0 is returned, which of course doesn't tell the caller much.

9.50.3.83 bool seq64::sequence::get_next_event (midibyte *status*, midibyte *cc*, long * *tick*, midibyte * *d0*, midibyte * *d1*, bool * *selected*)

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

Note the usage of event::is_desired_cc_or_not_cc(status, cc, *d0); Either we have a control change with the right CC or it's a different type of event.

9.50.3.84 bool seq64::sequence::get_next_event (midibyte * *a_status*, midibyte * *a_cc*)

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

9.50.3.85 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.

9.50.3.86 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.

9.50.3.87 void seq64::sequence::background_sequence (int *bs*) [inline]

Disabling the sequence number (setting it to SEQ64_SEQUENCE_LIMIT) is valid.

9.50.3.88 void seq64::sequence::put_event_on_bus (event * *ev*) [private]

Threadsafe

9.50.3.89 void seq64::sequence::set_trigger_offset (long *trigger_offset*) [private]

Threadsafe

9.50.3.90 void seq64::sequence::split_trigger (trigger & *trig*, long *splittick*) [private]

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.
<i>splittick</i>	The position just after where the original trigger will be truncated, and the new trigger begins.

9.50.3.91 void seq64::sequence::adjust_trigger_offsets_to_length (long *newlength*) [private]

Threadsafe

Might can get rid of this function?

9.50.3.92 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

9.50.3.93 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\(\)](#)!

9.50.4 Field Documentation

9.50.4.1 `int seq64::sequence::m_playing_notes[SEQ64_MIDI_NOTES_MAX] [private]`

It is used when muting, to shut off the notes that are playing.

9.50.4.2 `int seq64::sequence::m_seq_number [private]`

This number is set in the `perform::install_sequence()` function.

9.50.4.3 `long seq64::sequence::m_length [private]`

This value should be a power of two when used as a bar unit.

9.50.4.4 `midibyte seq64::sequence::m_musical_key [private]`

If the value is SEQ64_KEY_OF_C, then there is no musical key to be set.

9.50.4.5 `midibyte seq64::sequence::m_musical_scale [private]`

If the value is the enumeration value `c_scale_off`, then there is no musical scale to be set.

9.50.4.6 `long seq64::sequence::m_background_sequence [private]`

If the value is greater than `max_sequence()`, then there is no background sequence to be set.

9.50.4.7 `mutex seq64::sequence::m_mutex [mutable], [private]`

Made mutable for use in certain locked getter functions.

9.51 seq64::trigger Class Reference

This class hold a single trigger for a sequence object.

Public Member Functions

- `trigger ()`
Initializes the trigger structure.
- `bool operator< (const trigger &rhs)`
This operator compares only the m_tick_start members.
- `long tick_start () const`
'Getter' function for member m_tick_start
- `void tick_start (long s)`
'Setter' function for member m_tick_start
- `void increment_tick_start (long s)`
'Setter' function for member m_tick_start
- `void decrement_tick_start (long s)`
'Setter' function for member m_tick_start
- `long tick_end () const`

- 'Getter' function for member m_tick_end*
- void [tick_end](#) (long e)
- 'Setter' function for member m_tick_end*
- void [increment_tick_end](#) (long s)
- 'Setter' function for member m_tick_end*
- void [decrement_tick_end](#) (long s)
- 'Setter' function for member m_tick_end*
- long [offset](#) () const
- 'Getter' function for member m_offset*
- void [offset](#) (long o)
- 'Setter' function for member m_offset*
- void [increment_offset](#) (long s)
- 'Setter' function for member m_offset*
- void [decrement_offset](#) (long s)
- 'Setter' function for member m_offset*
- bool [selected](#) () const
- 'Getter' function for member m_selected*
- void [selected](#) (bool s)
- 'Setter' function for member m_selected*

Private Attributes

- long [m_tick_start](#)
Provides the starting tick for this trigger.
- long [m_tick_end](#)
Provides the ending tick for this trigger.
- long [m_offset](#)
Provides the offset for this trigger.
- bool [m_selected](#)
Indicates that the trigger is part of a selection.

9.51.1 Detailed Description

This class is used in playback, and is contained in the triggers class.

9.52 seq64::triggers Class Reference

The triggers class is a receptable the triggers that can be used with a sequence object.

Public Types

- typedef std::list< [trigger](#) > [List](#)
Exposes the triggers type, currently needed for [midi_container](#) only.

Public Member Functions

- [triggers](#) ([sequence](#) &parent)
Principal constructor.
- [~triggers](#) ()
A rote destructor.
- [triggers](#) & [operator=](#) (const [triggers](#) &rhs)
Principal assignment operator.
- void [set_ppqn](#) (int ppqn)
'Setter' function for member m_ppqn We have to set this value after construction for best safety.
- void [set_length](#) (int len)
'Setter' function for member m_length We have to set this value after construction for best safety.
- [List](#) & [triggerlist](#) ()
'Getter' function for member m_triggers
- void [push_undo](#) ()
Pushes the list-trigger into the trigger undo-list, then flags each item in the undo-list as unselected.
- void [pop_undo](#) ()
If the trigger undo-list has any items, the list-trigger is pushed into the redo list, the top of the undo-list is copied into the list-trigger, and then pops from the undo-list.
- void [print](#) (const std::string &seqname)
Prints a list of the currently-held triggers.
- bool [play](#) (long &start_tick, long &end_tick)
If playback-mode (live mode?) is in force, that is, if using in-triggers and on/off triggers, this function handles that kind of playback.
- void [add](#) (long tick, long len, long offset=0, bool adjustoffset=true)
Adds a trigger.
- void [adjust_offsets_to_length](#) (long newlen)
Adjusts trigger offsets to the length of ???, for all triggers, and undo triggers.
- void [split](#) (long tick)
Splits the first trigger that brackets the splittick parameter.
- void [split](#) ([trigger](#) &trig, long split_tick)
Splits the trigger given by the parameter into two triggers.
- void [grow](#) (long tick_from, long tick_to, long length)
Grows a trigger.
- void [remove](#) (long tick)
Deletes the first trigger that brackets the given tick from the trigger-list.
- bool [get_state](#) (long tick)
Checks the list of triggers against the given tick.
- bool [select](#) (long tick)
Checks the list of triggers against the given tick.
- bool [unselect](#) ()
Unselects all triggers.
- bool [intersect](#) (long position, long &start, long &end)
This function examines each trigger in the trigger list.
- void [remove_selected](#) ()
Deletes the first selected trigger that is found.
- void [copy_selected](#) ()
Copies the first selected trigger that is found.
- void [paste](#) ()
If there is a copied trigger, then this function grabs it from the trigger clipboard and adds it.
- bool [move_selected](#) (long tick, bool adjustoffset, int which=2)

- Moves selected triggers as per the given parameters.*

 - long [get_selected_start](#) ()
Gets the selected trigger's start tick.
 - long [get_selected_end](#) ()
Gets the selected trigger's end tick.
 - long [get_maximum](#) ()
Get the ending value of the last trigger in the trigger-list.
 - void [move](#) (long start_tick, long distance, bool direction)
Moves triggers in the trigger-list.
 - void [copy](#) (long start_tick, long distance)
Not sure what these diagrams are for yet.
 - void [clear](#) ()
Clears the whole list of triggers.
 - bool [next](#) (long *tick_on, long *tick_off, bool *selected, long *tick_offset)
'Getter' function for member m_trigger_offset
 - [trigger next_trigger](#) ()
Get the next trigger in the trigger list, and set the parameters based on that trigger.
 - void [reset_draw_trigger_marker](#) ()
Sets the draw-trigger iterator to the beginning of the trigger list.

Private Member Functions

- 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.

Private Attributes

- [sequence](#) & [m_parent](#)
Holds a reference to the parent sequence object that owns this trigger object.
- [List m_triggers](#)
This list holds the current pattern/triggers events.
- [trigger m_clipboard](#)
This item holds a single copied trigger, to be pasted later.
- [Stack m_undo_stack](#)
Handles the undo list for a series of operations on triggers.
- [Stack m_redo_stack](#)
Handles the redo list for a series of operations on triggers.
- [List::iterator m_iterator_play_trigger](#)
An iterator for cycling through the triggers during playback.
- [List::iterator m_iterator_draw_trigger](#)
An iterator for cycling through the triggers during drawing.
- bool [m_trigger_copied](#)
Set to true if there is an active trigger in the trigger clipboard.
- long [m_trigger_offset](#)
Offset.
- int [m_ppqn](#)
Holds the value of the PPQN from the parent sequence, for easy access.
- int [m_length](#)
Holds the value of the length from the parent sequence, for easy access.

9.52.1 Constructor & Destructor Documentation

9.52.1.1 seq64::triggers::triggers (*sequence* & *parent*)

Parameters

<i>parent</i>	The triggers object often needs to tell its parent sequence object what to do (such as stop playing).
---------------	---

9.52.2 Member Function Documentation

9.52.2.1 triggers & seq64::triggers::operator= (const triggers & rhs)

Follows the stock rules for such an operator, but does a little more than 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.

9.52.2.2 void seq64::triggers::set_length (int len) [inline]

Also, there a chance that the length of the parent might change from time to time. Currently, only the sequence constructor and midifile call this function.

9.52.2.3 bool seq64::triggers::play (long & start_tick, long & end_tick)

This is a new function for `sequence::play()` to call.

Parameters

<i>start_tick</i>	Provides the starting tick value, and returns the modified value as a side-effect.
<i>end_tick</i>	Provides the ending tick value, and returns the modified value as a side-effect.

Returns

Returns true if we're through playing the frame, and the caller should stop the playback.

9.52.2.4 void seq64::triggers::add (long tick, long len, long offset = 0, bool fixoffset = true)

What is this?

```

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

```

Parameters

<i>tick</i>	Provides the tick (pulse) time at which the trigger goes on.
<i>len</i>	Provides the length of the trigger. This value is actually calculated from the "on" value minus the "off" value read from the MIDI file.

<i>offset</i>	This value specifies the offset of the trigger. It is a feature of the <code>c_triggers_new</code> that <code>c_triggers</code> doesn't have. It is the third value in the trigger specification of the Sequencer64 MIDI file.
<i>fixoffset</i>	If true, the offset parameter is modified by <code>adjust_offset()</code> first. We think that basically makes sure it is positive.

9.52.2.5 void seq64::triggers::adjust_offsets_to_length (long *newlength*)

Parameters

<i>newlength</i>	
------------------	--

COMMON CODE?

COMMON CODE?

9.52.2.6 void seq64::triggers::split (long *splittick*)

This is the first trigger where *splittick* is greater than L and less than R.

Parameters

<i>splittick</i>	Provides the tick that must be bracketed for the split to be made.
------------------	--

9.52.2.7 void seq64::triggers::split (trigger & *trig*, long *splittick*)

The original trigger ends 1 tick before the *splittick* parameter, and the new trigger starts at *splittick* and ends where the original trigger ended.

Parameters

<i>trig</i>	Provides the original trigger, and also holds the changes made to that trigger as it is shortened.
<i>splittick</i>	The position just after where the original trigger will be truncated, and the new trigger begins.

9.52.2.8 void seq64::triggers::grow (long *tickfrom*, long *tickto*, long *len*)

This function looks for the first trigger where the *tickfrom* parameter is between the trigger's tick-start and tick-end values. If found then the trigger's start is moved back to *tickto*, if necessary, or the trigger's end is moved to *tickto* plus the length parameter, if necessary.

Then this new trigger is added, and the function breaks from the search loop.

Parameters

<i>tickfrom</i>	The desired from-value back which to expand the trigger, if necessary.
<i>tickto</i>	The desired to-value towards which to expand the trigger, if necessary.
<i>len</i>	The additional length to append to <i>tickto</i> for the check.

9.52.2.9 void seq64::triggers::remove (long *tick*)

Parameters

<i>tick</i>	Provides the tick to be examined.
-------------	-----------------------------------

9.52.2.10 bool seq64::triggers::get_state (long *tick*)

If any trigger is found to bracket that tick, then true is returned.

Parameters

<i>tick</i>	Provides the tick of interest.
-------------	--------------------------------

Returns

Returns true if a trigger is found that brackets the given tick.

9.52.2.11 bool seq64::triggers::select (long *tick*)

If any trigger is found to bracket that tick, then true is returned, and the trigger is marked as selected.

Parameters

<i>tick</i>	Provides the tick of interest.
-------------	--------------------------------

Returns

Returns true if a trigger is found that brackets the given tick.

9.52.2.12 bool seq64::triggers::unselect ()

Returns

Always returns false.

9.52.2.13 bool seq64::triggers::intersect (long *position*, long & *start*, long & *ender*)

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.

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>ender</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.

9.52.2.14 void seq64::triggers::paste ()

It pastes at the copy end.

9.52.2.15 `bool seq64::triggers::move_selected (long tick, bool fixoffset, int which = 2)`

```
mintick][0          1][maxtick
                2
```

Parameters

<i>which</i>	<p>Selects which movement will be done. This parameter has three possible values:</p> <ul style="list-style-type: none"> • 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.
--------------	--

Returns

Returns true if there was room to move. Otherwise, false is returned. We need this feature to support keystroke movement of a selected trigger in the perffroll window, and keep it from continually incremented when there can be no more movement. This causes moving the other direction to be delayed while the accumulating movement counter is used up. However, right now we can't rely on this result, and ignore it. There may be no way around this minor issue.

9.52.2.16 `long seq64::triggers::get_selected_start ()`

We guess this ends up selecting only one trigger, otherwise only the last selected one would effectively set the result.

Returns

Returns the `tick_start()` value of the last-selected trigger. If no triggers are selected, then -1 is returned.

9.52.2.17 `long seq64::triggers::get_selected_end ()`

Returns

Returns the `tick_end()` value of the last-selected trigger. If no triggers are selected, then -1 is returned.

9.52.2.18 `void seq64::triggers::copy (long starttick, long 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...

9.52.2.19 `bool seq64::triggers::next (long * tick_on, long * tick_off, bool * selected, long * offset)`

Get the next trigger in the trigger list, and set the parameters based on that trigger.

```
long get_trigger_offset () const { return m_trigger_offset; }
```

Todo It would be a bit simpler to simply return a trigger object, wouldn't it?

Parameters

<i>tick_on</i>	Return value for the retrieval of the starting tick for the trigger.
<i>tick_off</i>	Return value for the retrieval of the ending tick for the trigger.
<i>selected</i>	Return value for the retrieval of the is-selected flag for the trigger.
<i>offset</i>	Return value for the retrieval of the offset for the trigger.

Returns

Returns true if a trigger was found. If false, the caller cannot rely on the values returned through the return parameters.

Side-effect(s) The value of the `m_iterator_draw_trigger` member will be altered by this call, unless pointing to the end of the triggerlist, or if there are no triggers.

9.52.2.20 `trigger seq64::triggers::next_trigger ()`

Returns

Returns the next trigger. If there is none, a default trigger object is returned.

Side-effect(s) The value of the `m_iterator_draw_trigger` member will be altered by this call, unless pointing to the end of the triggerlist, or if there are no triggers.

9.52.2.21 `long seq64::triggers::adjust_offset (long offset)` `[private]`

Parameters

<i>offset</i>	Provides the offset, mod'ed against <code>m_length</code> , used to adjust the offset.
---------------	--

Returns

Returns the new offset. However, if `m_length` is 0, no change is made, and the original offset is returned.

9.52.3 Field Documentation

9.52.3.1 `int seq64::triggers::m_ppqn` `[private]`

This should not change, but we have to set it after construction, and so we provide a setter for it, `set_ppqn()`, called by the sequence constructor.

9.52.3.2 `int seq64::triggers::m_length` `[private]`

This might change, we're not yet sure.

9.53 seq64::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.
- 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.

9.53.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

9.53.2 Member Function Documentation

9.53.2.1 void seq64::user_instrument::set_defaults ()

Also invalidates the object.

9.53.2.2 int seq64::user_instrument::controller_max () const [inline]

Remember that the controller numbers for each MIDI instrument range from 0 to 127 (MIDI_CONTROLLER_MAX-1).

9.53.2.3 const std::string & seq64::user_instrument::controller_name (int *c*) const

Parameters

<i>c</i>	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.

9.53.2.4 bool seq64::user_instrument::controller_active (int *c*) const

Parameters

<i>c</i>	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.

9.53.2.5 void seq64::user_instrument::set_controller (int *c*, const std::string & *cname*, bool *isactive*)

Parameters

<i>c</i>	The index of the desired controller.
<i>cname</i>	The name of the controller to be set as the controller name.
<i>isactive</i>	A flag that indicates if the desired controller is active.

9.53.2.6 void seq64::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?

9.53.2.7 void seq64::user_instrument::copy_definitions (const user_instrument & *rhs*) [private]

Does not include the validity flag.

9.53.3 Field Documentation

9.53.3.1 `bool seq64::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.

9.53.3.2 `int seq64::user_instrument::m_controller_count` [private]

Often, the "user" configuration file has only a few out of the 128 assigned explicitly.

9.54 `seq64::user_instrument_t` Struct Reference

This structure corresponds to `[user-instrument-N]` definitions in the `~/ .seq24usr` or `~/ .config/sequencer64/seq64usr` file.

Data Fields

- `std::string instrument`
Provides the name of the "instrument" being supported.
- `std::string controllers` [SEQ64_MIDI_CONTROLLER_MAX]
Provides a list of up to 128 controllers (e.g.
- `bool controllers_active` [SEQ64_MIDI_CONTROLLER_MAX]
Provides a flag that indicates if each of up to 128 controller is active and supported.

9.54.1 Field Documentation

9.54.1.1 `std::string seq64::user_instrument_t::instrument`

Do not confuse "instrument" with "program" here. An "instrument" is most likely a hardware MIDI sound-box (though it could be a software synthesizer as well.

9.54.1.2 `std::string seq64::user_instrument_t::controllers`[SEQ64_MIDI_CONTROLLER_MAX]

"Modulation"). If a controller isn't present, or if General MIDI is in force, this name might be empty.

9.54.1.3 `bool seq64::user_instrument_t::controllers_active`[SEQ64_MIDI_CONTROLLER_MAX]

If false, it might be an unsupported controller or a General MIDI device.

9.55 `seq64::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.
- 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 SEQ64_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.

9.55.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

9.55.2 Member Function Documentation

9.55.2.1 void seq64::user_midi_bus::set_defaults ()

Also invalidates the object. All 16 of the channels are set to SEQ64_GM_INSTRUMENT_FLAG (-1).

9.55.2.2 `int seq64::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.

9.55.2.3 `int seq64::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).

9.55.2.4 `int seq64::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 SEQ64_GM_INSTRUMENT_FLAG (-1) is returned.

9.55.2.5 `void seq64::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.

9.55.2.6 `void seq64::user_midi_bus::copy_definitions (const user_midi_bus & rhs) [private]`

Does not include the validity flag.

9.55.3 Field Documentation

9.55.3.1 `bool seq64::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.

9.55.3.2 `int seq64::user_midi_bus::m_channel_count [private]`

Often, the "user" configuration file has only a few out of the 16 assigned explicitly.

9.56 seq64::user_midi_bus_t Struct Reference

This structure corresponds to `[user-midi-bus-0]` definitions in the `~/.seq24usr` ("user") file (`~/.config/sequencer64/sequencer64 usr` in the latest version of the application).

Data Fields

- `std::string alias`
Provides the user's desired name for the MIDI bus.
- `int instrument [SEQ64_MIDI_BUS_CHANNEL_MAX]`
Provides an implicit list of MIDI channels from 0 to 15 (1 to 16) and the "instrument" number assigned to each channel.

9.56.1 Field Documentation

9.56.1.1 `std::string seq64::user_midi_bus_t::alias`

For example, "2x2 A" for some kind of MIDI card or USB MIDI cable. If manual-alsa-ports is enabled, this could be something like "[0] seq24 0", and that is what should be shown in that case.

9.56.1.2 `int seq64::user_midi_bus_t::instrument[SEQ64_MIDI_BUS_CHANNEL_MAX]`

Note that the "instrument" is not a MIDI program number. Instead, it is the number associated with a "user-instrument" section in the "user" configuration file.

9.57 seq64::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 ()`
Scale factor for PPQN.
- `user_settings (const user_settings &rhs)`
Copy constructor.
- `user_settings & operator= (const user_settings &rhs)`
Principal assignment operator.
- `void set_defaults ()`
Sets the default values.
- `void normalize ()`
Calculate the derived values from the already-set values.
- `void set_globals () const`
Copies the current values of the member variables into their corresponding global variables.
- `void get_globals ()`
Copies the current values of the global variables into their corresponding member variables.
- `bool add_bus (const std::string &alias)`
Adds a user bus to the container, but only does so if the name parameter is not empty.
- `bool add_instrument (const std::string &instname)`
Adds a user instrument to the container, but only does so if the name parameter is not empty.
- `const user_midi_bus & bus (int index)`
'Getter' function for member Unlike the non-const version this function is public.
- `const user_instrument & instrument (int index)`
'Getter' function for member Unlike the non-const version this function is public.
- `int bus_count () const`
'Getter' function for member m_midi_buses.size()

- void [set_bus_instrument](#) (int index, int channel, int instrum)
'Getter' function for member m_midi_buses[index].instrument[channel] Currently this function is used, in the [userfile::parse\(\)](#) function.
- int [bus_instrument](#) (int buss, int channel)
'Getter' function for member m_midi_buses[buss].instrument[channel]
- const std::string & [bus_name](#) (int buss)
'Getter' function for member m_midi_buses[buss].name
- int [instrument_count](#) () const
'Getter' function for member m_instruments.size()
- void [set_instrument_controllers](#) (int index, int cc, const std::string &ccname, bool isactive)
'Setter' function for member m_midi_instrument_defs[index].controllers, controllers_active
- const std::string & [instrument_name](#) (int instrum)
'Getter' function for member m_instruments[instrument].instrument (name of instrument)
- const std::string & [instrument_name](#) (int buss, int channel)
Gets the correct instrument number from the buss and channel, and then looks up the name of the instrument.
- bool [instrument_controller_active](#) (int instrum, int cc)
'Getter' function for member m_instruments[instrument].controllers_active[controller]
- bool [controller_active](#) (int buss, int channel, int cc)
A convenience function so that the caller doesn't have to get the instrument number from the [bus_instrument\(\)](#) member function.
- const std::string & [instrument_controller_name](#) (int instrum, int cc)
'Getter' function for member m_instruments[instrument].controllers_active[controller]
- const std::string & [controller_name](#) (int buss, int channel, int cc)
'Getter' function for member m_instruments[instrument].controllers_active[controller] A convenience function so that the caller doesn't have to get the instrument number from the [bus_instrument\(\)](#) member function.
- int [grid_style](#) () const
'Getter' function for member m_grid_style Checks for normal style.
- bool [grid_is_normal](#) () const
'Getter' function for member m_grid_style Checks for normal style.
- bool [grid_is_white](#) () const
'Getter' function for member m_grid_style Checks for the white style.
- bool [grid_is_black](#) () const
'Getter' function for member m_grid_style Checks for the black style.
- int [grid_brackets](#) () const
'Getter' function for member m_grid_brackets
- 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, dependent member
- int [gmute_tracks](#) () const
'Getter' function for member m_gmute_tracks, dependent member
- int [max_sets](#) () const
'Getter' function for member m_max_sets
- int [max_sequence](#) () const
'Getter' function for member m_max_sequence, dependent member
- int [text_x](#) () const
'Getter' function for member m_text_x, not user modifiable, not saved
- int [text_y](#) () const
'Getter' function for member m_text_y, not user modifiable, not saved

- int [seqchars_x](#) () const
'Getter' function for member m_seqchars_x, not user modifiable, not saved
- int [seqchars_y](#) () const
'Getter' function for member m_seqchars_y, not user modifiable, not saved
- int [seqarea_x](#) () const
'Getter' function for member m_seqarea_x, not user modifiable, not saved
- int [seqarea_y](#) () const
'Getter' function for member m_seqarea_y, not user modifiable, not saved
- int [seqarea_seq_x](#) () const
'Getter' function for member m_seqarea_seq_x, not user modifiable, not saved
- int [seqarea_seq_y](#) () const
'Getter' function for member m_seqarea_seq_y, not user modifiable, not saved
- int [mainwid_border](#) () const
'Getter' function for member m_mainwid_border
- int [mainwid_spacing](#) () const
'Getter' function for member m_mainwid_spacing
- int [mainwid_x](#) () const
'Getter' function for member m_mainwid_x, dependent member
- int [mainwid_y](#) () const
'Getter' function for member m_mainwid_y, dependent member
- int [control_height](#) () const
'Getter' function for member m_control_height
- int [zoom](#) () const
'Getter' function for member m_current_zoom
- void [zoom](#) (int value)
'Setter' function for member m_current_zoom This value is not modified unless the value parameter is between 1 and 32, inclusive.
- bool [global_seq_feature](#) () const
'Getter' function for member m_global_seq_feature_save
- void [global_seq_feature](#) (bool flag)
'Setter' function for member m_global_seq_feature_save
- int [seqedit_scale](#) () const
'Getter' function for member m_seqedit_scale
- void [seqedit_scale](#) (int scale)
'Setter' function for member m_seqedit_scale
- int [seqedit_key](#) () const
'Getter' function for member m_seqedit_key
- void [seqedit_key](#) (int key)
'Setter' function for member m_seqedit_key
- int [seqedit_bgsequence](#) () const
'Getter' function for member m_seqedit_bgsequence
- void [seqedit_bgsequence](#) (int seqnum)
'Setter' function for member m_seqedit_bgsequence Note that SEQ64_IS_LEGAL_SEQUENCE() allows the SEQ64_SEQUENCE_LIMIT (0x800 = 2048) value, to turn off the use of a background sequence.
- bool [use_new_font](#) () const
'Getter' function for member m_use_new_font
- void [use_new_font](#) (bool flag)
'Setter' function for member m_use_new_font
- bool [save_user_config](#) () const
'Getter' function for member m_save_user_config
- void [save_user_config](#) (bool flag)

- *'Setter' function for member m_save_user_config*
- int `midi_ppqn` () const
 - *'Getter' function for member m_midi_ppqn*
- int `midi_beats_per_bar` () const
 - *'Getter' function for member m_midi_beats_per_measure*
- int `midi_beats_per_minute` () const
 - *'Getter' function for member m_midi_beats_per_minute*
- int `midi_beat_width` () const
 - *'Getter' function for member m_midi_beat_width*
- char `midi_buss_override` () const
 - *'Getter' function for member m_midi_buss_override*
- int `min_zoom` () const
 - *'Getter' function for member mc_min_zoom*
- int `max_zoom` () const
 - *'Getter' function for member mc_max_zoom*
- int `baseline_ppqn` () const
 - *'Getter' function for member mc_baseline_ppqn*
- void `midi_ppqn` (int ppqn)
 - *'Setter' function for member m_midi_ppqn This value can be set from 96 to 960 (this upper limit will be determined by what Sequencer64 can actually handle).*
- void `midi_buss_override` (char buss)
 - *'Setter' function for member m_midi_buss_override This value can be set from 0 to 31.*

Protected Member Functions

- void `grid_brackets` (int thickness)
 - *'Getter' function for member m_grid_brackets*
- void `grid_style` (int gridstyle)
 - *'Setter' function for member m_grid_style*
- 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_max_sets This value is not modified unless the value parameter is between 32 and 64, inclusive.*
- void `text_x` (int value)
 - *'Setter' function for member m_text_x This value is not modified unless the value parameter is between 6 and 6, inclusive.*
- 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](#) ()
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.
- void [midi_beats_per_bar](#) (int beatsperbar)
'Setter' function for member m_midi_beats_per_measure This value can be set from 1 to 16.
- void [midi_beats_per_minute](#) (int beatsperminute)
'Setter' function for member m_midi_beats_minute This value can be set from 20 to 500.
- void [midi_beat_width](#) (int beatwidth)
'Setter' function for member m_midi_beatwidth This value can be set to any power of 2 in the range from 1 to 16.

Private Types

- enum [mainwid_grid_style_t](#) {
 [grid_style_normal](#),
 [grid_style_white](#),
 [grid_style_black](#),
 [grid_style_max](#) }
- typedef std::vector< [user_midi_bus](#) > [Busses](#)
 [*user-midi-bus-definitions*]
- typedef std::vector< [user_instrument](#) > [Instruments](#)
 [*user-instrument-definitions*]

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.

- [mainwid_grid_style_t m_grid_style](#)
[user-interface-settings]
- [int m_grid_brackets](#)
Specify drawing brackets (like the old Seq24) or a solid box.
- [int m_mainwnd_rows](#)
Number of rows in the Patterns Panel.
- [int m_mainwnd_cols](#)
Number of columns in the Patterns Panel.
- [int m_max_sets](#)
Maximum number of screen sets that can be supported.
- [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_current_zoom](#)
Provides the initial zoom value, in units of.
- [bool m_global_seq_feature_save](#)
If true, this value provide a bit of backward-compatibility with the global key/scale/background-sequence persistence feature.
- [int m_seqedit_scale](#)
Replaces seqedit::m_initial_scale as the repository for the scale to apply when a sequence is loaded into the sequence editor.
- [int m_seqedit_key](#)
Replaces seqedit::m_initial_key as the repository for the key to apply when a sequence is loaded into the sequence editor.
- [int m_seqedit_bgsequence](#)
Replaces seqedit::m_initial_sequence as the repository for the background sequence to apply when a sequence is loaded into the sequence editor.
- [bool m_use_new_font](#)
Sets the usage of the font.
- [int m_text_x](#)
Constants for the mainwid class.
- [int m_seqchars_x](#)
Constants for the mainwid class.
- [int m_midi_ppqn](#)
Provides the universal PPQN setting for the duration of this setting.
- [int m_midi_beats_per_measure](#)
Provides the universal and unambiguous MIDI value for beats per measure, also called "beats per bar" (BPB).
- [int m_midi_beats_per_minute](#)
Provides the universal and unambiguous MIDI value for beats per minute (BPM).
- [int m_midi_beat_width](#)
Provides the universal MIDI value for beats width (BW).
- [char m_midi_buss_override](#)
Provides a universal override of the buss number for all sequences, for the purpose of convenience of of testing.
- [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 supported, which is m_seqs_in_set squared, or 1024.
- [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_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_x](#)
The width of the main pattern/sequence grid, in pixels.
- bool [m_save_user_config](#)
Provides a temporary variable that can be set from the command line to cause the "user" state to be saved into the "user" configuration file.
- const int [mc_min_zoom](#)
Provides the minimum zoom value, currently a constant.
- const int [mc_max_zoom](#)
Provides the maximum zoom value, currently a constant.
- const int [mc_baseline_ppqn](#)
Permanent storage for the baseline, default PPQN used by Seq24.

9.57.1 Detailed Description

These settings will eventually be made part of the "user" settings file.

9.57.2 Member Typedef Documentation

9.57.2.1 `typedef std::vector<user_midi_bus> seq64::user_settings::Busses` `[private]`

Internal type for the container of [user_midi_bus](#) objects. Sorry about the "confusion" about "bus" versus "buss". See Google for arguments about it.

9.57.2.2 `typedef std::vector<user_instrument> seq64::user_settings::Instruments` `[private]`

Internal type for the container of [user_instrument](#) objects.

9.57.3 Member Enumeration Documentation

9.57.3.1 `enum seq64::user_settings::mainwid_grid_style_t` `[private]`

Enumerator

grid_style_normal Provides a setting to control the overall style of grid-drawing for the pattern slots in mainwid. These values can be specified in the [user-interface-settings] section of the "user" configuration file.

The grid background color is the normal background color for the current GTK theme. The box is drawn with brackets on either side.

grid_style_white The grid background color is white. This style better fits displaying the white-on-black sequence numbers. The box is drawn with brackets on either side.

grid_style_black The grid background color is black.

grid_style_max Marks the end of the list, and is an illegal value.

9.57.4 Constructor & Destructor Documentation

9.57.4.1 seq64::user_settings::user_settings ()

Default constructor.

Should this be a float (6 significant digits) or two integers for scaling?

9.57.5 Member Function Documentation

9.57.5.1 void seq64::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!

9.57.5.2 void seq64::user_settings::set_globals () const

Should be called at initialization, and after settings are read from the "user" configuration file.

DO NOT PUT ANY GLOBALS HERE UNTIL THEIR EFFECTS HAVE BEEN TESTED!!!!

9.57.5.3 void seq64::user_settings::get_globals ()

Should be called before settings are written to the "user" configuration file.

9.57.5.4 const user_midi_bus& seq64::user_settings::bus (int *index*) [inline]

Cannot append the const specifier.

9.57.5.5 const user_instrument& seq64::user_settings::instrument (int *index*) [inline]

Cannot append the const specifier.

9.57.5.6 bool seq64::user_settings::controller_active (int *buss*, int *channel*, int *cc*) [inline]

It also has a shorter name.

9.57.5.7 const std::string& seq64::user_settings::controller_name (int *buss*, int *channel*, int *cc*) [inline]

It also has a shorter name.

9.57.5.8 void seq64::user_settings::zoom (int *value*)

The default value is 2.

9.57.5.9 void seq64::user_settings::mainwnd_rows (int *value*) [protected]

The default value is 4. Dependent values are recalculated after the assignment.

9.57.5.10 void seq64::user_settings::mainwnd_cols (int *value*) [protected]

The default value is 8. Dependent values are recalculated after the assignment.

9.57.5.11 void seq64::user_settings::max_sets (int *value*) [protected]

The default value is 32. Dependent values are recalculated after the assignment.

9.57.5.12 void seq64::user_settings::text_x (int *value*) [protected]

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.

9.57.5.13 void seq64::user_settings::text_y (int *value*) [protected]

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.

9.57.5.14 void seq64::user_settings::mainwid_border (int *value*) [protected]

The default value is 0. Dependent values are recalculated after the assignment.

9.57.5.15 void seq64::user_settings::mainwid_spacing (int *value*) [protected]

The default value is 2. Dependent values are recalculated after the assignment.

9.57.5.16 void seq64::user_settings::control_height (int *value*) [protected]

The default value is 0. Dependent values are recalculated after the assignment.

9.57.5.17 void seq64::user_settings::dump_summary () [protected]

Does its work only if PLATFORM_DEBUG and SEQ64_USE_DEBUG_OUTPUT are defined. Only enabled in emergencies :-D.

9.57.5.18 void seq64::user_settings::midi_ppqn (int *value*)

The default value is 192. Dependent values may be recalculated after the assignment.

9.57.5.19 void seq64::user_settings::midi_buss_override (char *buss*)

The default value is -1, which means that there is no buss override. It provides a way to override the buss number for smallish MIDI files. It replaces the buss-number read from the file. This option is turned on by the `-bus` option, and is merely a convenience feature for the quick previewing of a tune. (It's called "developer laziness".)

9.57.5.20 void seq64::user_settings::midi_beats_per_bar (int *value*) [protected]

The default value is 4.

9.57.5.21 void seq64::user_settings::midi_beats_per_minute (int *value*) [protected]

The default value is 120.

9.57.5.22 `void seq64::user_settings::midi_beat_width (int bw)` [protected]

The default value is 4.

9.57.5.23 `user_midi_bus & seq64::user_settings::private_bus (int index)` [private]

This invalid object has an empty alias, and all the instrument numbers are -1.

9.57.5.24 `user_instrument & seq64::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.

9.57.6 Field Documentation

9.57.6.1 `Busses seq64::user_settings::m_midi_buses` [private]

Since this object is a vector, its size is adjustable.

9.57.6.2 `Instruments seq64::user_settings::m_instruments` [private]

The size is adjustable, and grows as objects are added.

9.57.6.3 `mainwid_grid_style_t seq64::user_settings::m_grid_style` [private]

Specifies the current grid style.

9.57.6.4 `int seq64::user_settings::m_grid_brackets` [private]

0 = no brackets, 1 and above is the thickness of the brackets. 1 is the normal thickness of the brackets, 2 is a two-pixel thickness, and so on.

9.57.6.5 `int seq64::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". We would like to be able to change this value from 4 to 8, and maybe allow the values of 5, 6, and 7 as well. But if we could just get 8 working, then well would Sequencer64 deserve the 64 in its name.

9.57.6.6 `int seq64::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".

9.57.6.7 `int seq64::user_settings::m_max_sets` [private]

Basically, that the number of times the Patterns Panel can be filled. 32 sets can be created. Although this value is part of the "user" configuration file, it is likely that it will never change. Rather, the number of sequences per set would change. We'll see.

9.57.6.8 `int seq64::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.

9.57.6.9 `int seq64::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.

9.57.6.10 `bool seq64::user_settings::m_global_seq_feature_save` `[private]`

In this feature, applying one of these three changes to a sequence causes them to also be applied to sequences that are subsequently opened for editing. However, we improve on this feature by allowing the changes to be saved in the global, proprietary part of the saved MIDI file.

If false, the user can still save the key/scale/background-sequence values with each individual sequence, so they can be different.

This value will be true by default, unless changed in the "user" configuration file.

9.57.6.11 `int seq64::user_settings::m_seqedit_scale` `[private]`

Its default value is `c_scale_off`. Although this value is now stored in the [user_settings](#) class, it always comes from the currently loaded MIDI file, if present. If `m_global_seq_feature_save` is true, this variable is stored in the "proprietary" track at the end of the file, under the control tag `c_musicscale`, and will be applied to any sequence that is edited. If `m_global_seq_feature_save` is false, this variable is stored, if used, in the meta-data for the sequence to which it applies, and, again, is tagged with the control tag `c_musicscale`.

9.57.6.12 `int seq64::user_settings::m_seqedit_key` `[private]`

Its default value is `SEQ64_KEY_OF_C`. Although this value is now stored in the [user_settings](#) class, it always comes from the currently loaded MIDI file, if present. If `m_global_seq_feature_save` is true, this variable is stored in the "proprietary" track at the end of the file, under the control tag `c_musickey`, and will be applied to any sequence that is edited. If `m_global_seq_feature_save` is false, this variable is stored, if used, in the meta-data for the sequence to which it applies, and, again, is tagged with the control tag `c_musickey`.

9.57.6.13 `int seq64::user_settings::m_seqedit_bgsequence` `[private]`

Its default value is `SEQ64_SEQUENCE_LIMIT`. Although this value is now stored in the [user_settings](#) class, it always comes from the currently loaded MIDI file, if present. If `m_global_seq_feature_save` is true, this variable is stored, if it has a valid (but not "legal") value, in the "proprietary" track at the end of the file, under the control tag `c_backsequence`, and will be applied to any sequence that is edited. If `m_global_seq_feature_save` is false, this variable is stored, if used, in the meta-data for the sequence to which it applies, and, again, is tagged with the control tag `c_backsequence`.

9.57.6.14 `bool seq64::user_settings::m_use_new_font` `[private]`

By default, in normal mode, the new font is used. In legacy mode, the old font is used.

9.57.6.15 `int seq64::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).

We have added black-on-yellow and yellow-on-black versions of the fonts, to support the highlighting of pattern boxes if they are empty of actual MIDI events.

We have also added a set of four new font files that are roughly the same size, and are treated as the same size, but look smooth and less like a DOS-era font.

The font module does not use these values directly, but does define some similar variables that differ slightly between the two styles of font. There are a lot of tricks and hard-wired places to fix before further work can be done with fonts in Sequencer64.

9.57.6.16 `int seq64::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.

9.57.6.17 `int seq64::user_settings::m_midi_ppqn` [private]

This variable replaces the global `ppqn`. The default value of this setting is 192 parts-per-quarter-note (PPQN). There is still a lot of work to get a different PPQN to work properly in speed of playback, scaling of the user interface, and other issues. Note that this value can be changed by the still-experimental `-ppqn` option. There is one remaining trace of the global, though: `DEFAULT_PPQN`.

9.57.6.18 `int seq64::user_settings::m_midi_beats_per_measure` [private]

This variable will replace the global beats per measure. The default value of this variable is `DEFAULT_BEATS_PER_MEASURE` (4). For external access, we will call this value "beats per bar", abbreviate it "BPB", and use "bpb" in any accessor function names. Now, although it applies to the whole session, we should be able to continue seq24's tradition of allowing each sequence to have its own time signature. Also, there are a number of places where the number 4 appears and looks like it might be a hardwired BPB value, either for MIDI purposes or for drawing the piano-roll grids. So we might need a couple different versions of this variable.

9.57.6.19 `int seq64::user_settings::m_midi_beats_per_minute` [private]

This variable will replace the global beats per minute. The default value of this variable is `DEFAULT_BPM` (120). This variable should apply to the whole session; there's probably no way to support a different tempo for each sequence. But we shall see. For external access, we will call this value "beats per minute", abbreviate it "BPM", and use "bpm" in any accessor function names.

9.57.6.20 `int seq64::user_settings::m_midi_beat_width` [private]

This variable will replace the global `beat_width`. The default value of this variable is `DEFAULT_BEAT_WIDTH` (4). Now, although it applies to the whole session, we should be able to continue seq24's tradition of allowing each sequence to have its own time signature. Also, there are a number of places where the number 4 appears and looks like it might be a hardwired BW value, either for MIDI purposes or for drawing the user-interface. So we might need a couple different versions of this variable. For external access, we will call this value "beat width", abbreviate it "BW", and use "bw" in any accessor function names.

9.57.6.21 `char seq64::user_settings::m_midi_buss_override` [private]

This variable replaces the global buss-override variable, and is set via the command-line option `-bus`.

9.57.6.22 `int seq64::user_settings::m_seqs_in_set` `[private]`

This value is $4 \times 8 = 32$ by default.

Warning

Currently implicit/explicit in a number of the "rc" file and [rc_settings](#). Would probably want the left 32 or the first 32 items in the main window only to be subject to keystroke control. This value is calculated by the [normalize\(\)](#) function, and is *not* part of the "user" configuration file.

9.57.6.23 `int seq64::user_settings::m_gmute_tracks` `[private]`

This value is *not* part of the "user" configuration file; it is calculated by the [normalize\(\)](#) function.

9.57.6.24 `int seq64::user_settings::m_max_sequence` `[private]`

It is a derived value, and not stored in the "user" file.

```
m_max_sequence = m_seqs_in_set * m_max_sets;
```

9.57.6.25 `int seq64::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.

9.57.6.26 `int seq64::user_settings::m_seqarea_seq_x` `[private]`

These are used only in the `mainwid` module.

9.57.6.27 `int seq64::user_settings::m_mainwid_x` `[private]`

Affected by the `m_mainwid_border` and `m_mainwid_spacing` values.

```
c_mainwid_x =
(
    (c_seqarea_x + c_mainwid_spacing) * c_mainwnd_cols -
    c_mainwid_spacing + c_mainwid_border * 2
);
```

9.57.6.28 `bool seq64::user_settings::m_save_user_config` `[private]`

Normally, this state is not saved. It is not saved because there is currently no user-interface for editing it, and because it can pick up some command-line options, and it is not right to have them written to the "user" configuration file.

(The "rc" configuration file is a different case, having historically always been saved, and having a number of command-line options, such as JACK settings that should generally be permanent on a given system.)

Anyway, this flag can be set by the `-user-save` option. This setting is never saved. But note that, if no "user" configuration file is found, it is then saved anyway.

9.57.6.29 `const int seq64::user_settings::mc_min_zoom` `[private]`

It's value is 1.

9.57.6.30 `const int seq64::user_settings::mc_max_zoom` `[private]`

It's value is 32.

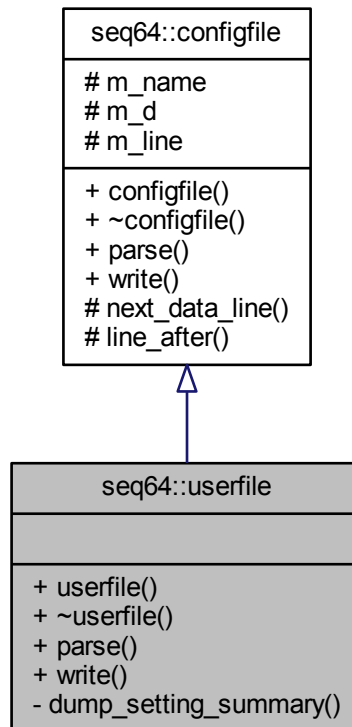
9.57.6.31 `const int seq64::user_settings::mc_baseline_ppqn` `[private]`

This value is necessary in order to keep user-interface elements stable when different PPQNs are used. It is set to `DEFAULT_PPQN`.

9.58 seq64::userfile Class Reference

Supports the user's `~/ .config/sequencer64/sequencer64.usr` and `~/ .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.

Private Member Functions

- void [dump_setting_summary](#) ()

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.

Additional Inherited Members

9.58.1 Member Function Documentation

9.58.1.1 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](#).

9.58.1.2 bool seq64::userfile::write (const [perform](#) & *a_perf*) [virtual]

Parameters

<i>a_perf</i>	The performance object, currently unused.
---------------	---

Implements [seq64::configfile](#).

9.58.1.3 void seq64::userfile::dump_setting_summary () [private]

Does work only if PLATFORM_DEBUG is defined; see the [user_settings](#) class.

Index

- ~event
 - seq64::event, [34](#)
- ~keys_perform
 - seq64::keys_perform, [65](#)
- ~keys_perform_gtk2
 - seq64::keys_perform_gtk2, [69](#)
- ~perfedit
 - seq64::perfedit, [119](#)
- ~perform
 - seq64::perform, [130](#)
- ~perfroll
 - seq64::perfroll, [149](#)
- ~seqmenu
 - seq64::seqmenu, [186](#)
- about_dialog
 - seq64::mainwnd, [89](#)
- add
 - seq64::event_list, [41](#)
 - seq64::triggers, [220](#)
- add_event
 - seq64::sequence, [205](#), [210](#)
- add_long
 - seq64::midi_container, [93](#)
- add_note
 - seq64::sequence, [210](#)
- add_sequence
 - seq64::perform, [132](#)
- add_trigger
 - seq64::sequence, [205](#)
- add_variable
 - seq64::midi_container, [93](#)
- adj_callback_bpm
 - seq64::mainwnd, [89](#)
- adj_callback_ss
 - seq64::mainwnd, [89](#)
- adjust_offset
 - seq64::triggers, [225](#)
- adjust_offsets_to_length
 - seq64::triggers, [221](#)
- adjust_trigger_offsets_to_length
 - seq64::sequence, [214](#)
- alias
 - seq64::user_midi_bus_t, [231](#)
- all_notes_off
 - seq64::perform, [135](#)
- any_selected_notes
 - seq64::event_list, [42](#)
- append_sysex
 - seq64::event, [37](#)
- apply_length
 - seq64::seqedit, [171](#)
- BLACK
 - seq64::font, [44](#)
- BLACK_ON_YELLOW
 - seq64::font, [44](#)
- background_sequence
 - seq64::sequence, [214](#)
- bus
 - seq64::user_settings, [238](#)
- Busses
 - seq64::user_settings, [237](#)
- calculate_base_sizes
 - seq64::mainwid, [82](#)
- change_event_data_range
 - seq64::sequence, [210](#)
- channel_count
 - seq64::user_midi_bus, [229](#)
- channel_max
 - seq64::user_midi_bus, [230](#)
- CharList
 - seq64::midi_list, [95](#)
- clamp_track
 - seq64::perform, [143](#)
- clear_all
 - seq64::perform, [131](#)
- clear_sequence_triggers
 - seq64::perform, [132](#)
- clear_triggers
 - seq64::sequence, [209](#)
- clear_window
 - seq64::gui_drawingarea_gtk2, [51](#)
- click
 - seq64::click, [29](#)
- clock
 - seq64::midibus, [100](#)
- collapse
 - seq64::perfedit, [120](#)
- Color
 - seq64::font, [44](#)
- configfile
 - seq64::configfile, [31](#)
- continue_from
 - seq64::midibus, [100](#)
- control_height
 - seq64::user_settings, [239](#)
- controller_active
 - seq64::user_instrument, [227](#)

- seq64::user_settings, 238
- controller_max
 - seq64::user_instrument, 227
- controller_name
 - seq64::user_instrument, 227
 - seq64::user_settings, 238
- controllers
 - seq64::user_instrument_t, 228
- controllers_active
 - seq64::user_instrument_t, 228
- convert_t
 - seq64::seqevent, 178
- convert_tn
 - seq64::seqroll, 192
- convert_x
 - seq64::perfroll, 150
 - seq64::seqevent, 178
- convert_xy
 - seq64::perfroll, 150
- copy
 - seq64::perfedit, 120
 - seq64::triggers, 224
- copy_definitions
 - seq64::user_instrument, 227
 - seq64::user_midi_bus, 230
- copy_selected
 - seq64::sequence, 210
- copy_triggers
 - seq64::perform, 133
 - seq64::sequence, 209
- count
 - seq64::event_list, 41
- count_selected_events
 - seq64::event_list, 42
- create_menus
 - seq64::seqedit, 172
- decrement_beats_per_minute
 - seq64::perform, 139
- decrement_selected
 - seq64::sequence, 211
- deinit_in
 - seq64::midibus, 99
- del_trigger
 - seq64::sequence, 205
- delete_sequence
 - seq64::perform, 132
- do_action
 - seq64::seqedit, 173
- draw_background
 - seq64::seqevent, 177
- draw_drawable_row
 - seq64::perfroll, 151
- draw_events_on
 - seq64::seqevent, 178
 - seq64::seqroll, 192
- draw_events_on_pixmap
 - seq64::seqroll, 191
- draw_key
 - seq64::seqkeys, 183
- draw_line
 - seq64::gui_drawingarea_gtk2, 51–53
- draw_line_on_pixmap
 - seq64::gui_drawingarea_gtk2, 51, 52
- draw_marker_on_sequence
 - seq64::mainwid, 80
- draw_normal_rectangle_on_pixmap
 - seq64::gui_drawingarea_gtk2, 55
- draw_pixmap_on_window
 - seq64::seqevent, 177
- draw_rectangle
 - seq64::gui_drawingarea_gtk2, 53–55
- draw_rectangle_on_pixmap
 - seq64::gui_drawingarea_gtk2, 55
- draw_sequence
 - seq64::perfnames, 123
- draw_sequence_on
 - seq64::perfroll, 151
- draw_sequence_on_pixmap
 - seq64::mainwid, 81
- draw_sequence_pixmap_on_window
 - seq64::mainwid, 81
- draw_sequences_on_pixmap
 - seq64::mainwid, 81
- drop_event
 - seq64::seqevent, 178
- dump_setting_summary
 - seq64::userfile, 245
- dump_summary
 - seq64::user_settings, 239
- e_deselect
 - seq64::sequence, 201
- e_is_selected
 - seq64::sequence, 201
- e_remove_one
 - seq64::sequence, 201
- e_select
 - seq64::sequence, 201
- e_select_one
 - seq64::sequence, 201
- e_toggle_selection
 - seq64::sequence, 201
- e_would_select
 - seq64::sequence, 201
- edit_callback_notepad
 - seq64::mainwnd, 89
- errdump
 - seq64::midifile, 111
- error_message
 - seq64::jack_assistant, 61
- event_count
 - seq64::sequence, 202
- event_key
 - seq64::event_list::event_key, 38
- event_list
 - seq64::event_list, 41
- events

- seq64::keybindentry, 63
- expand
 - seq64::perfedit, 120
- file_import_dialog
 - seq64::mainwnd, 88
- fill
 - seq64::midi_container, 92
- fill_background_pixmap
 - seq64::perfroll, 150
- fill_container
 - seq64::sequence, 213
- get
 - seq64::midi_container, 93
 - seq64::midi_list, 95
 - seq64::midi_vector, 97
- get_beat_width
 - seq64::sequence, 202
- get_data
 - seq64::event, 37
- get_globals
 - seq64::user_settings, 238
- get_group_mute_state
 - seq64::perform, 138
- get_highest_note_event
 - seq64::sequence, 213
- get_keys
 - seq64::keys_perform, 66
- get_lowest_note_event
 - seq64::sequence, 213
- get_max_trigger
 - seq64::perform, 133
 - seq64::sequence, 209
- get_measures
 - seq64::seqedit, 171
- get_name
 - seq64::sequence, 203
- get_next_event
 - seq64::sequence, 213
- get_next_note_event
 - seq64::sequence, 213
- get_num_selected_events
 - seq64::sequence, 210
- get_num_selected_notes
 - seq64::sequence, 210
- get_rank
 - seq64::event, 37
- get_screen_set_notepad
 - seq64::perform, 134
- get_selected_end
 - seq64::triggers, 224
- get_selected_start
 - seq64::triggers, 224
- get_sequence
 - seq64::perform, 137
- get_state
 - seq64::triggers, 222
- get_trigger_state
 - seq64::sequence, 205
- grid_style_black
 - seq64::user_settings, 237
- grid_style_max
 - seq64::user_settings, 237
- grid_style_normal
 - seq64::user_settings, 237
- grid_style_white
 - seq64::user_settings, 237
- groups
 - seq64::keybindentry, 63
- grow
 - seq64::perfedit, 119
 - seq64::triggers, 221
- grow_selected
 - seq64::sequence, 211
- grow_trigger
 - seq64::sequence, 205
- gui_assistant
 - seq64::gui_assistant, 47
- gui_palette_gtk2
 - seq64::gui_palette_gtk2, 57
- gui_window_gtk2
 - seq64::gui_window_gtk2, 59
- handle_config
 - seq64::lash, 73
- handle_event
 - seq64::lash, 73
- handle_midi_control
 - seq64::perform, 134
- handle_motion_key
 - seq64::Seq24PerfInput, 160
- highlight
 - seq64::perform, 139
- home_config_directory
 - seq64::rc_settings, 158
- idle_progress
 - seq64::maintime, 76
- idle_redraw
 - seq64::seqdata, 165
 - seq64::sequevent, 177
- increment_beats_per_minute
 - seq64::perform, 139
- increment_selected
 - seq64::sequence, 211
- info_message
 - seq64::jack_assistant, 61
- init
 - seq64::font, 44
 - seq64::jack_assistant, 60
 - seq64::lash, 73
 - seq64::perform, 131
- init_before_show
 - seq64::perfedit, 119
 - seq64::perfroll, 150
- init_clock
 - seq64::midibus, 100

- init_in
 - seq64::midibus, 99
- init_in_sub
 - seq64::midibus, 99
- init_jack
 - seq64::perform, 131
- init_out
 - seq64::midibus, 99
- init_out_sub
 - seq64::midibus, 99
- inner_start
 - seq64::perform, 143
- install_sequence
 - seq64::perform, 143
- instrument
 - seq64::user_instrument_t, 228
 - seq64::user_midi_bus, 230
 - seq64::user_midi_bus_t, 231
 - seq64::user_settings, 238
- Instruments
 - seq64::user_settings, 237
- intersect
 - seq64::triggers, 222
- intersect_events
 - seq64::sequence, 208
- intersect_notes
 - seq64::sequence, 207
- intersect_triggers
 - seq64::sequence, 207
- is_active
 - seq64::perform, 137
- is_channel_msg
 - seq64::event, 35
- is_desired_cc_or_not_cc
 - seq64::event, 35
- is_dirty_edit
 - seq64::perform, 136
 - seq64::sequence, 204
- is_dirty_main
 - seq64::perform, 136
 - seq64::sequence, 204
- is_dirty_names
 - seq64::perform, 136
 - seq64::sequence, 204
- is_dirty_perf
 - seq64::perform, 136
 - seq64::sequence, 204
- is_letter
 - seq64::keystroke, 71
- is_midi_control_valid
 - seq64::perform, 142
- is_modified
 - seq64::perform, 142
- is_mseq_valid
 - seq64::perform, 142
- is_one_byte_msg
 - seq64::event, 35
- is_screenset_valid
 - seq64::perform, 142
- is_seq_valid
 - seq64::perform, 142
- is_sysex_special_id
 - seq64::midifile, 111
- is_two_byte_msg
 - seq64::event, 35
- jack_assistant
 - seq64::jack_assistant, 60
- jack_shutdown
 - seq64::jack_assistant, 61
- jack_sync_callback
 - seq64::jack_assistant, 61
 - seq64::perform, 143
- jack_timebase_callback
 - seq64::jack_assistant, 61
- key_name
 - seq64::keys_perform, 66
 - seq64::keys_perform_gtk2, 69
- keybindentry
 - seq64::keybindentry, 63
- keystroke
 - seq64::keystroke, 70, 71
- lash
 - seq64::lash, 72
- lash_timeout_connect
 - seq64::gui_assistant_gtk2, 48
- launch_input_thread
 - seq64::perform, 131
- launch_output_thread
 - seq64::perform, 131
- line_after
 - seq64::configfile, 31
- link_new
 - seq64::event_list, 42
 - seq64::sequence, 212
- location
 - seq64::keybindentry, 63
- m_b_on_y_pixmap
 - seq64::font, 45
- m_background
 - seq64::gui_drawingarea_gtk2, 56
- m_background_sequence
 - seq64::sequence, 215
- m_bar_width
 - seq64::maintime, 77
- m_beat_width
 - seq64::maintime, 77
- m_black_pixmap
 - seq64::font, 45
- m_box_height
 - seq64::maintime, 77
- m_box_width
 - seq64::maintime, 77
- m_bpm

- seq64::perfedit, 120
- m_button
 - seq64::click, 29
- m_bw
 - seq64::perfedit, 120
- m_channel_count
 - seq64::user_midi_bus, 230
- m_char_list
 - seq64::midifile, 112
- m_char_w
 - seq64::perfnames, 123
- m_clip_mask
 - seq64::font, 45
- m_control_height
 - seq64::user_settings, 241
- m_controller_count
 - seq64::user_instrument, 228
- m_data
 - seq64::event, 37
 - seq64::midifile, 112
- m_disable_reported
 - seq64::midifile, 112
- m_drop_x
 - seq64::gui_drawingarea_gtk2, 56
- m_error_message
 - seq64::midifile, 112
- m_file_size
 - seq64::midifile, 112
- m_flash_height
 - seq64::maintime, 77
- m_flash_width
 - seq64::maintime, 77
- m_font_h
 - seq64::font, 44
- m_font_w
 - seq64::font, 44
- m_foreground
 - seq64::gui_drawingarea_gtk2, 56
- m_fruity_interaction
 - seq64::perffroll, 152
- m_global_seq_feature_save
 - seq64::user_settings, 241
- m_gmute_tracks
 - seq64::user_settings, 243
- m_grid_brackets
 - seq64::user_settings, 240
- m_grid_style
 - seq64::user_settings, 240
- m_has_link
 - seq64::event, 37
- m_initial_snap
 - seq64::seqedit, 173
- m_instruments
 - seq64::user_settings, 240
- m_is_modified
 - seq64::perform, 145
- m_is_press
 - seq64::keystroke, 71
- m_is_valid
 - seq64::user_instrument, 228
 - seq64::user_midi_bus, 230
- m_key
 - seq64::keybindentry, 63
 - seq64::keystroke, 71
- m_key_bpm_up
 - seq64::keys_perform, 67
- m_key_show_ui_sequence_number
 - seq64::keys_perform, 67
- m_left_marker_tick
 - seq64::perftime, 155
- m_left_tick
 - seq64::perform, 144
- m_length
 - seq64::sequence, 215
 - seq64::triggers, 225
- m_line
 - seq64::configfile, 31
- m_main_wid
 - seq64::mainwnd, 91
- m_mainperf
 - seq64::gui_drawingarea_gtk2, 56
- m_mainwid_border
 - seq64::user_settings, 240
- m_mainwid_x
 - seq64::user_settings, 243
- m_mainwnd_cols
 - seq64::user_settings, 240
- m_mainwnd_rows
 - seq64::user_settings, 240
- m_max_sequence
 - seq64::user_settings, 243
- m_max_sets
 - seq64::perform, 145
 - seq64::user_settings, 240
- m_measure_length
 - seq64::perftime, 155
- m_midi_beat_width
 - seq64::user_settings, 242
- m_midi_beats_per_measure
 - seq64::user_settings, 242
- m_midi_beats_per_minute
 - seq64::user_settings, 242
- m_midi_buses
 - seq64::user_settings, 240
- m_midi_buss_override
 - seq64::user_settings, 242
- m_midi_ppqn
 - seq64::user_settings, 242
- m_modifier
 - seq64::click, 29
 - seq64::keystroke, 71
- m_musical_key
 - seq64::sequence, 215
- m_musical_scale
 - seq64::sequence, 215
- m_mutex

- seq64::sequence, 215
- m_namebox_w
 - seq64::perfnames, 123
- m_names_chars
 - seq64::perfnames, 123
- m_names_y
 - seq64::perfnames, 123
- m_new_format
 - seq64::midifile, 112
- m_notebook
 - seq64::options, 113
- m_number_h
 - seq64::seqdata, 166
- m_number_offset_y
 - seq64::seqdata, 166
- m_number_w
 - seq64::seqdata, 166
- m_one_measure
 - seq64::perform, 144
- m_pixmap
 - seq64::font, 44
 - seq64::gui_drawingarea_gtk2, 56
- m_playback_mode
 - seq64::perform, 144
- m_playing_notes
 - seq64::sequence, 215
- m_playscreen_offset
 - seq64::perform, 144
- m_pos
 - seq64::midifile, 112
- m_ppqn
 - seq64::maintime, 77
 - seq64::triggers, 225
- m_rank
 - seq64::event_list::event_key, 39
- m_redraw_ms
 - seq64::perfedit, 120
- m_right_marker_tick
 - seq64::perftime, 156
- m_right_tick
 - seq64::perform, 144
- m_save_user_config
 - seq64::user_settings, 243
- m_screenset_offset
 - seq64::mainwid, 84
- m_screenset_slots
 - seq64::mainwid, 84
- m_seq_number
 - seq64::sequence, 215
- m_seqarea_seq_x
 - seq64::user_settings, 243
- m_seqarea_x
 - seq64::user_settings, 243
- m_seqchars_x
 - seq64::user_settings, 242
- m_seqedit
 - seq64::seqmenu, 187
- m_seqedit_bgsequence
 - seq64::user_settings, 241
- m_seqedit_key
 - seq64::user_settings, 241
- m_seqedit_scale
 - seq64::user_settings, 241
- m_seqs
 - seq64::perform, 144
- m_seqs_in_set
 - seq64::perform, 144
 - seq64::user_settings, 242
- m_sequence_count
 - seq64::perform, 145
- m_sequence_max
 - seq64::perform, 145
- m_setbox_w
 - seq64::perfnames, 123
- m_sigpipe
 - seq64::mainwnd, 91
- m_smf0_channels
 - seq64::midifile, 112
- m_smf0_channels_count
 - seq64::midifile, 112
- m_spinbutton_load_offset
 - seq64::mainwnd, 91
- m_starting_tick
 - seq64::perform, 144
- m_status
 - seq64::event, 37
- m_sysex
 - seq64::event, 37
- m_text_x
 - seq64::user_settings, 241
- m_timestamp
 - seq64::event_list::event_key, 39
- m_use_new_font
 - seq64::user_settings, 241
- m_white_pixmap
 - seq64::font, 45
- m_window
 - seq64::gui_drawingarea_gtk2, 56
- m_window_x
 - seq64::gui_drawingarea_gtk2, 56
 - seq64::gui_window_gtk2, 59
- m_x
 - seq64::click, 29
- m_xy_offset
 - seq64::perfnames, 124
- m_y
 - seq64::click, 29
- m_y_on_b_pixmap
 - seq64::font, 45
- maintime
 - seq64::maintime, 75
- mainwid
 - seq64::mainwid, 80
- mainwid_border
 - seq64::user_settings, 239
- mainwid_grid_style_t

- seq64::user_settings, 237
- mainwid_spacing
 - seq64::user_settings, 239
- mainwnd
 - seq64::mainwnd, 88
- mainwnd_cols
 - seq64::user_settings, 238
- mainwnd_key_event
 - seq64::perform, 140
- mainwnd_rows
 - seq64::user_settings, 238
- mark_out_of_range
 - seq64::event_list, 42
- mark_selected
 - seq64::sequence, 212
- max_sets
 - seq64::user_settings, 238
- mc_baseline_ppqn
 - seq64::user_settings, 244
- mc_max_zoom
 - seq64::user_settings, 243
- mc_min_zoom
 - seq64::user_settings, 243
- merge
 - seq64::event_list, 41
- midi_beat_width
 - seq64::user_settings, 239
- midi_beats_per_bar
 - seq64::user_settings, 239
- midi_beats_per_minute
 - seq64::user_settings, 239
- midi_buss_override
 - seq64::user_settings, 239
- midi_control_off
 - seq64::perform, 134
- midi_control_on
 - seq64::perform, 133
- midi_control_toggle
 - seq64::perform, 133
- midi_ppqn
 - seq64::user_settings, 239
- midifile
 - seq64::midifile, 103
- mod_timestamp
 - seq64::event, 36
- modify
 - seq64::perform, 131
- move_selected
 - seq64::triggers, 222
- move_selected_triggers_to
 - seq64::sequence, 208
- move_triggers
 - seq64::perform, 133
 - seq64::sequence, 209
- name_change_callback
 - seq64::seqedit, 172
- new_file
 - seq64::mainwnd, 90
- new_sequence
 - seq64::perform, 131
- next
 - seq64::triggers, 225
- next_data_line
 - seq64::configfile, 31
- next_trigger
 - seq64::triggers, 225
- off_playing_notes
 - seq64::sequence, 212
- off_queued
 - seq64::sequence, 203
- on_button_press_event
 - seq64::Seq24PerfInput, 160
 - seq64::Seq24SeqEventInput, 161
 - seq64::Seq24SeqRollInput, 162
 - seq64::mainwid, 82
 - seq64::perfroll, 151
 - seq64::seqdata, 166
 - seq64::seqevent, 178
 - seq64::seqkeys, 183
 - seq64::seqtime, 194
- on_button_release_event
 - seq64::Seq24PerfInput, 160
 - seq64::Seq24SeqEventInput, 161
 - seq64::Seq24SeqRollInput, 162
 - seq64::mainwid, 83
 - seq64::perfroll, 151
 - seq64::seqdata, 166
 - seq64::seqevent, 179
 - seq64::seqkeys, 183
 - seq64::seqtime, 194
- on_delete_event
 - seq64::mainwnd, 90
 - seq64::seqedit, 173
- on_enter_notify_event
 - seq64::seqkeys, 183
- on_expose_event
 - seq64::maintime, 77
 - seq64::mainwid, 82
 - seq64::perfnames, 123
 - seq64::perfroll, 151
 - seq64::perftime, 155
- on_focus_in_event
 - seq64::mainwid, 83
- on_focus_out_event
 - seq64::mainwid, 83
- on_grouplearnchange
 - seq64::mainwnd, 90
- on_key_press_event
 - seq64::keybindentry, 63
 - seq64::mainwnd, 90
 - seq64::perfroll, 151
 - seq64::seqevent, 179
 - seq64::seqroll, 192
- on_key_release_event
 - seq64::mainwnd, 90
- on_leave_notify_event

- seq64::seqkeys, 183
- on_motion_notify_event
 - seq64::Seq24PerfInput, 160
 - seq64::Seq24SeqEventInput, 161
 - seq64::mainwid, 83
 - seq64::seqdata, 166
 - seq64::seqevent, 179
 - seq64::seqkeys, 183
- on_realize
 - seq64::gui_drawingarea_gtk2, 56
 - seq64::maintime, 77
 - seq64::mainwid, 82
 - seq64::perfnames, 123
 - seq64::perfroll, 151
 - seq64::perftime, 155
 - seq64::seqdata, 166
 - seq64::seqevent, 178
 - seq64::seqkeys, 183
- on_scroll_event
 - seq64::seqdata, 166
 - seq64::seqroll, 192
- on_size_allocate
 - seq64::perfnames, 123
- open_file
 - seq64::mainwnd, 88
- open_performance_edit
 - seq64::mainwnd, 89
- operator<
 - seq64::event, 34
 - seq64::event_list::event_key, 38
- operator=
 - seq64::click, 29
 - seq64::event_list, 41
 - seq64::keystroke, 71
 - seq64::sequence, 202
 - seq64::triggers, 220
- output
 - seq64::jack_assistant, 61
- output_func
 - seq64::perform, 138
- parse
 - seq64::midifile, 104
 - seq64::optionsfile, 114
 - seq64::userfile, 245
- parse_prop_header
 - seq64::midifile, 106
- parse_proprietary_track
 - seq64::midifile, 107
- parse_smf_0
 - seq64::midifile, 106
- parse_smf_1
 - seq64::midifile, 106
- paste
 - seq64::triggers, 222
- paste_selected
 - seq64::sequence, 210
- paste_trigger
 - seq64::sequence, 208
- perfeddit
 - seq64::perfeddit, 119
- perfnames
 - seq64::perfnames, 123
- perform
 - seq64::perform, 130
- perfroll_key_event
 - seq64::perform, 140
- perftime
 - seq64::perftime, 155
- play
 - seq64::midibus, 100
 - seq64::perform, 137
 - seq64::sequence, 204
 - seq64::triggers, 220
- play_note_off
 - seq64::sequence, 212
- play_note_on
 - seq64::sequence, 212
- pop_redo
 - seq64::sequence, 202
- pop_undo
 - seq64::sequence, 202
- popup_event_menu
 - seq64::seqedit, 172
- popup_menu
 - seq64::seqmenu, 186
- popup_midibus_menu
 - seq64::seqedit, 173
- popup_sequence_menu
 - seq64::seqedit, 173
- popup_tool_menu
 - seq64::seqedit, 173
- position
 - seq64::jack_assistant, 60
 - seq64::midi_container, 93
- ppqn
 - seq64::mainwnd, 88
 - seq64::midifile, 106
- print
 - seq64::sequence, 204
- print_triggers
 - seq64::sequence, 204
- private_bus
 - seq64::user_settings, 240
- private_instrument
 - seq64::user_settings, 240
- process_events
 - seq64::lash, 73
- prop_item_size
 - seq64::midifile, 111
- push_trigger_undo
 - seq64::perform, 133
 - seq64::sequence, 202
- push_undo
 - seq64::sequence, 202
- put
 - seq64::midi_container, 93

- seq64::midi_list, 95
- seq64::midi_vector, 97
- put_event_on_bus
 - seq64::sequence, 214
- read_byte_array
 - seq64::midifile, 108
- read_long
 - seq64::midifile, 108
- read_seq_number
 - seq64::midifile, 109
- read_track_name
 - seq64::midifile, 109
- read_varinum
 - seq64::midifile, 108
- redraw
 - seq64::mainwid, 82
 - seq64::seqdata, 165
- redraw_events
 - seq64::seqroll, 191
- remove
 - seq64::sequence, 214
 - seq64::triggers, 221
- remove_marked
 - seq64::sequence, 212
- render_string
 - seq64::gui_drawingarea_gtk2, 53
- render_string_on_drawable
 - seq64::font, 44
- render_string_on_pixmap
 - seq64::gui_drawingarea_gtk2, 53
- reset
 - seq64::seqdata, 165
 - seq64::seqroll, 191
- reset_draw_marker
 - seq64::sequence, 213
- reset_draw_trigger_marker
 - seq64::sequence, 213
- reset_sequences
 - seq64::perform, 137
- save_file
 - seq64::mainwnd, 90
- save_playing_state
 - seq64::perform, 139
- select
 - seq64::triggers, 222
- select_action_e
 - seq64::sequence, 201
- select_all
 - seq64::sequence, 210
- select_events
 - seq64::sequence, 209
- select_group_mute
 - seq64::perform, 135
- select_mute_group
 - seq64::perform, 135
- select_note_events
 - seq64::sequence, 209
- select_trigger
 - seq64::sequence, 207
- selected_trigger_end
 - seq64::sequence, 208
- selected_trigger_start
 - seq64::sequence, 208
- seq64::AbstractPerfInput, 27
- seq64::Seq24PerfInput, 158
 - handle_motion_key, 160
 - on_button_press_event, 160
 - on_button_release_event, 160
 - on_motion_notify_event, 160
 - set_adding, 160
- seq64::Seq24SeqEventInput, 161
 - on_button_press_event, 161
 - on_button_release_event, 161
 - on_motion_notify_event, 161
 - set_adding, 161
- seq64::Seq24SeqRollInput, 162
 - on_button_press_event, 162
 - on_button_release_event, 162
 - set_adding, 162
- seq64::click, 27
 - click, 29
 - m_button, 29
 - m_modifier, 29
 - m_x, 29
 - m_y, 29
 - operator=, 29
- seq64::configfile, 30
 - configfile, 31
 - line_after, 31
 - m_line, 31
 - next_data_line, 31
- seq64::event, 31
 - ~event, 34
 - append_sysex, 37
 - get_data, 37
 - get_rank, 37
 - is_channel_msg, 35
 - is_desired_cc_or_not_cc, 35
 - is_one_byte_msg, 35
 - is_two_byte_msg, 35
 - m_data, 37
 - m_has_link, 37
 - m_status, 37
 - m_sysex, 37
 - mod_timestamp, 36
 - operator<, 34
 - set_data, 36
 - set_status, 36
- seq64::event_list, 39
 - add, 41
 - any_selected_notes, 42
 - count, 41
 - count_selected_events, 42
 - event_list, 41
 - link_new, 42

- mark_out_of_range, 42
- merge, 41
- operator=, 41
- verify_and_link, 42
- seq64::event_list::event_key, 38
 - event_key, 38
 - m_rank, 39
 - m_timestamp, 39
 - operator<, 38
- seq64::font, 42
 - BLACK, 44
 - BLACK_ON_YELLOW, 44
 - Color, 44
 - init, 44
 - m_b_on_y_pixmap, 45
 - m_black_pixmap, 45
 - m_clip_mask, 45
 - m_font_h, 44
 - m_font_w, 44
 - m_pixmap, 44
 - m_white_pixmap, 45
 - m_y_on_b_pixmap, 45
 - render_string_on_drawable, 44
 - WHITE, 44
 - YELLOW_ON_BLACK, 44
- seq64::gui_assistant, 45
 - gui_assistant, 47
- seq64::gui_assistant_gtk2, 47
 - lash_timeout_connect, 48
 - sm_internal_keys, 48
- seq64::gui_drawingarea_gtk2, 48
 - clear_window, 51
 - draw_line, 51–53
 - draw_line_on_pixmap, 51, 52
 - draw_normal_rectangle_on_pixmap, 55
 - draw_rectangle, 53–55
 - draw_rectangle_on_pixmap, 55
 - m_background, 56
 - m_drop_x, 56
 - m_foreground, 56
 - m_mainperf, 56
 - m_pixmap, 56
 - m_window, 56
 - m_window_x, 56
 - on_realize, 56
 - render_string, 53
 - render_string_on_pixmap, 53
 - set_line, 51
- seq64::gui_drawingarea_gtk2::rect, 158
- seq64::gui_palette_gtk2, 56
 - gui_palette_gtk2, 57
- seq64::gui_window_gtk2, 58
 - gui_window_gtk2, 59
 - m_window_x, 59
- seq64::jack_assistant, 59
 - error_message, 61
 - info_message, 61
 - init, 60
 - jack_assistant, 60
 - jack_shutdown, 61
 - jack_sync_callback, 61
 - jack_timebase_callback, 61
 - output, 61
 - position, 60
 - stop, 60
- seq64::jack_scratchpad, 62
- seq64::keybindentry, 62
 - events, 63
 - groups, 63
 - keybindentry, 63
 - location, 63
 - m_key, 63
 - on_key_press_event, 63
 - set, 63
 - type, 63
- seq64::keys_perform, 63
 - ~keys_perform, 65
 - get_keys, 66
 - key_name, 66
 - m_key_bpm_up, 67
 - m_key_show_ui_sequence_number, 67
 - set_all_key_events, 66
 - set_all_key_groups, 66
 - set_key_event, 66
 - set_key_group, 67
 - set_keys, 65
 - show_ui_sequence_key, 66
 - show_ui_sequence_number, 66
- seq64::keys_perform_gtk2, 67
 - ~keys_perform_gtk2, 69
 - key_name, 69
 - set_all_key_events, 69
 - set_all_key_groups, 69
- seq64::keys_perform_transfer, 69
- seq64::keystroke, 69
 - is_letter, 71
 - keystroke, 70, 71
 - m_is_press, 71
 - m_key, 71
 - m_modifier, 71
 - operator=, 71
- seq64::lash, 72
 - handle_config, 73
 - handle_event, 73
 - init, 73
 - lash, 72
 - process_events, 73
 - set_alsa_client_id, 73
- seq64::maintime, 73
 - idle_progress, 76
 - m_bar_width, 77
 - m_beat_width, 77
 - m_box_height, 77
 - m_box_width, 77
 - m_flash_height, 77
 - m_flash_width, 77

- m_ppqn, 77
 - maintime, 75
 - on_expose_event, 77
 - on_realize, 77
- seq64::mainwid, 78
 - calculate_base_sizes, 82
 - draw_marker_on_sequence, 80
 - draw_sequence_on_pixmap, 81
 - draw_sequence_pixmap_on_window, 81
 - draw_sequences_on_pixmap, 81
 - m_screenset_offset, 84
 - m_screenset_slots, 84
 - mainwid, 80
 - on_button_press_event, 82
 - on_button_release_event, 83
 - on_expose_event, 82
 - on_focus_in_event, 83
 - on_focus_out_event, 83
 - on_motion_notify_event, 83
 - on_realize, 82
 - redraw, 82
 - seq_from_xy, 81
 - set_screenset, 80
 - timeout, 82
 - update_markers, 80
 - update_sequence_on_window, 80
 - valid_sequence, 81
- seq64::mainwnd, 84
 - about_dialog, 89
 - adj_callback_bpm, 89
 - adj_callback_ss, 89
 - edit_callback_notepad, 89
 - file_import_dialog, 88
 - m_main_wid, 91
 - m_sigpipe, 91
 - m_spinbutton_load_offset, 91
 - mainwnd, 88
 - new_file, 90
 - on_delete_event, 90
 - on_grouplearnchange, 90
 - on_key_press_event, 90
 - on_key_release_event, 90
 - open_file, 88
 - open_performance_edit, 89
 - ppqn, 88
 - save_file, 90
 - signal_action, 90
 - timer_callback, 89
 - update_window_title, 89
- seq64::midi_container, 91
 - add_long, 93
 - add_variable, 93
 - fill, 92
 - get, 93
 - position, 93
 - put, 93
- seq64::midi_list, 93
 - CharList, 95
 - get, 95
 - put, 95
- seq64::midi_vector, 95
 - get, 97
 - put, 97
- seq64::midibus, 97
 - clock, 100
 - continue_from, 100
 - deinit_in, 99
 - init_clock, 100
 - init_in, 99
 - init_in_sub, 99
 - init_out, 99
 - init_out_sub, 99
 - play, 100
 - set_input, 100
 - sysex, 100
- seq64::midifile, 100
 - errdump, 111
 - is_sysex_special_id, 111
 - m_char_list, 112
 - m_data, 112
 - m_disable_reported, 112
 - m_error_message, 112
 - m_file_size, 112
 - m_new_format, 112
 - m_pos, 112
 - m_smf0_channels, 112
 - m_smf0_channels_count, 112
 - midifile, 103
 - parse, 104
 - parse_prop_header, 106
 - parse_proprietary_track, 107
 - parse_smf_0, 106
 - parse_smf_1, 106
 - ppqn, 106
 - prop_item_size, 111
 - read_byte_array, 108
 - read_long, 108
 - read_seq_number, 109
 - read_track_name, 109
 - read_varinum, 108
 - smf0_increment, 106
 - varinum_size, 110
 - write, 104
 - write_byte, 108
 - write_long, 108
 - write_prop_header, 109
 - write_proprietary_track, 110
 - write_seq_number, 109
 - write_short, 108
 - write_track_name, 109
 - write_varinum, 108
- seq64::options, 112
 - m_notebook, 113
- seq64::optionsfile, 113
 - parse, 114
 - write, 116

- seq64::perfedit, 116
 - ~perfedit, 119
 - collapse, 120
 - copy, 120
 - expand, 120
 - grow, 119
 - init_before_show, 119
 - m_bpm, 120
 - m_bw, 120
 - m_redraw_ms, 120
 - perfedit, 119
 - set_beat_width, 119
 - set_beats_per_bar, 119
 - set_guides, 119
 - start_playing, 120
 - stop_playing, 120
 - timeout, 120
 - undo, 120
- seq64::perfnames, 121
 - draw_sequence, 123
 - m_char_w, 123
 - m_namebox_w, 123
 - m_names_chars, 123
 - m_names_y, 123
 - m_setbox_w, 123
 - m_xy_offset, 124
 - on_expose_event, 123
 - on_realize, 123
 - on_size_allocate, 123
 - perfnames, 123
- seq64::perform, 124
 - ~perform, 130
 - add_sequence, 132
 - all_notes_off, 135
 - clamp_track, 143
 - clear_all, 131
 - clear_sequence_triggers, 132
 - copy_triggers, 133
 - decrement_beats_per_minute, 139
 - delete_sequence, 132
 - get_group_mute_state, 138
 - get_max_trigger, 133
 - get_screen_set_notepad, 134
 - get_sequence, 137
 - handle_midi_control, 134
 - highlight, 139
 - increment_beats_per_minute, 139
 - init, 131
 - init_jack, 131
 - inner_start, 143
 - install_sequence, 143
 - is_active, 137
 - is_dirty_edit, 136
 - is_dirty_main, 136
 - is_dirty_names, 136
 - is_dirty_perf, 136
 - is_midi_control_valid, 142
 - is_modified, 142
 - is_mseq_valid, 142
 - is_screenset_valid, 142
 - is_seq_valid, 142
 - jack_sync_callback, 143
 - launch_input_thread, 131
 - launch_output_thread, 131
 - m_is_modified, 145
 - m_left_tick, 144
 - m_max_sets, 145
 - m_one_measure, 144
 - m_playback_mode, 144
 - m_playscreen_offset, 144
 - m_right_tick, 144
 - m_seqs, 144
 - m_seqs_in_set, 144
 - m_sequence_count, 145
 - m_sequence_max, 145
 - m_starting_tick, 144
 - mainwnd_key_event, 140
 - midi_control_off, 134
 - midi_control_on, 133
 - midi_control_toggle, 133
 - modify, 131
 - move_triggers, 133
 - new_sequence, 131
 - output_func, 138
 - perform, 130
 - perfroll_key_event, 140
 - play, 137
 - push_trigger_undo, 133
 - reset_sequences, 137
 - save_playing_state, 139
 - select_group_mute, 135
 - select_mute_group, 135
 - seq_in_playing_screen, 140
 - sequence_count, 131
 - sequence_label, 139
 - sequence_playing_off, 138
 - sequence_playing_on, 138
 - set_active, 135
 - set_beats_per_minute, 137
 - set_group_mute_state, 138
 - set_input_bus, 140
 - set_key_event, 143
 - set_key_group, 143
 - set_left_tick, 132
 - set_offset, 139
 - set_orig_ticks, 137
 - set_playing_screenset, 135
 - set_right_tick, 133
 - set_screen_set_notepad, 134
 - set_screenset, 134
 - set_sequence_control_status, 138
 - set_was_active, 136
 - show_ui_sequence_key, 139
 - show_ui_sequence_number, 139
 - sm_mc_dummy, 144
 - start, 135

- start_playing, 139
- stop, 135
- unset_mode_group_learn, 135
- unset_sequence_control_status, 138
- seq64::performcallback, 145
- seq64::perfroll, 146
 - ~perfroll, 149
 - convert_x, 150
 - convert_xy, 150
 - draw_drawable_row, 151
 - draw_sequence_on, 151
 - fill_background_pixmap, 150
 - init_before_show, 150
 - m_fruity_interaction, 152
 - on_button_press_event, 151
 - on_button_release_event, 151
 - on_expose_event, 151
 - on_key_press_event, 151
 - on_realize, 151
 - set_guides, 149
 - set_ppqn, 150
 - snap_x, 150
 - start_playing, 150
 - stop_playing, 151
 - update_sizes, 150
- seq64::perftime, 152
 - m_left_marker_tick, 155
 - m_measure_length, 155
 - m_right_marker_tick, 156
 - on_expose_event, 155
 - on_realize, 155
 - perftime, 155
 - set_guides, 155
- seq64::rc_settings, 156
 - home_config_directory, 158
- seq64::rect, 158
- seq64::seqdata, 162
 - idle_redraw, 165
 - m_number_h, 166
 - m_number_offset_y, 166
 - m_number_w, 166
 - on_button_press_event, 166
 - on_button_release_event, 166
 - on_motion_notify_event, 166
 - on_realize, 166
 - on_scroll_event, 166
 - redraw, 165
 - reset, 165
 - seqdata, 165
 - set_data_type, 165
 - set_zoom, 165
 - update_sizes, 165
 - xy_to_rect, 165
- seq64::seqedit, 167
 - apply_length, 171
 - create_menus, 172
 - do_action, 173
 - get_measures, 171
 - m_initial_snap, 173
 - name_change_callback, 172
 - on_delete_event, 173
 - popup_event_menu, 172
 - popup_midibus_menu, 173
 - popup_sequence_menu, 173
 - popup_tool_menu, 173
 - seqedit, 170
 - set_background_sequence, 172
 - set_data_type, 172
 - set_key, 171
 - set_measures, 171
 - set_midi_bus, 171
 - set_midi_channel, 171
 - set_note_length, 170
 - set_scale, 171
 - set_snap, 170
 - set_zoom, 170
- seq64::seqevent, 174
 - convert_t, 178
 - convert_x, 178
 - draw_background, 177
 - draw_events_on, 178
 - draw_pixmap_on_window, 177
 - drop_event, 178
 - idle_redraw, 177
 - on_button_press_event, 178
 - on_button_release_event, 179
 - on_key_press_event, 179
 - on_motion_notify_event, 179
 - on_realize, 178
 - set_data_type, 177
 - set_snap, 177
 - snap_x, 178
 - start_paste, 178
 - update_sizes, 177
 - x_to_w, 178
- seq64::seqkeys, 179
 - draw_key, 183
 - on_button_press_event, 183
 - on_button_release_event, 183
 - on_enter_notify_event, 183
 - on_leave_notify_event, 183
 - on_motion_notify_event, 183
 - on_realize, 183
 - set_hint_state, 182
- seq64::seqmenu, 184
 - ~seqmenu, 186
 - m_seqedit, 187
 - popup_menu, 186
 - seq_clear_perf, 186
 - seq_copy, 186
 - seq_cut, 186
 - seq_edit, 186
 - seq_paste, 186
 - seqmenu, 186
- seq64::seqroll, 187
 - convert_tn, 192

- draw_events_on, 192
- draw_events_on_pixmap, 191
- on_key_press_event, 192
- on_scroll_event, 192
- redraw_events, 191
- reset, 191
- set_background_sequence, 191
- set_data_type, 191
- snap_x, 192
- update_background, 191
- update_sizes, 191
- seq64::seqtime, 192
 - on_button_press_event, 194
 - on_button_release_event, 194
- seq64::sequence, 194
 - add_event, 205, 210
 - add_note, 210
 - add_trigger, 205
 - adjust_trigger_offsets_to_length, 214
 - background_sequence, 214
 - change_event_data_range, 210
 - clear_triggers, 209
 - copy_selected, 210
 - copy_triggers, 209
 - decrement_selected, 211
 - del_trigger, 205
 - e_deselect, 201
 - e_is_selected, 201
 - e_remove_one, 201
 - e_select, 201
 - e_select_one, 201
 - e_toggle_selection, 201
 - e_would_select, 201
 - event_count, 202
 - fill_container, 213
 - get_beat_width, 202
 - get_highest_note_event, 213
 - get_lowest_note_event, 213
 - get_max_trigger, 209
 - get_name, 203
 - get_next_event, 213
 - get_next_note_event, 213
 - get_num_selected_events, 210
 - get_num_selected_notes, 210
 - get_trigger_state, 205
 - grow_selected, 211
 - grow_trigger, 205
 - increment_selected, 211
 - intersect_events, 208
 - intersect_notes, 207
 - intersect_triggers, 207
 - is_dirty_edit, 204
 - is_dirty_main, 204
 - is_dirty_names, 204
 - is_dirty_perf, 204
 - link_new, 212
 - m_background_sequence, 215
 - m_length, 215
 - m_musical_key, 215
 - m_musical_scale, 215
 - m_mutex, 215
 - m_playing_notes, 215
 - m_seq_number, 215
 - mark_selected, 212
 - move_selected_triggers_to, 208
 - move_triggers, 209
 - off_playing_notes, 212
 - off_queued, 203
 - operator=, 202
 - paste_selected, 210
 - paste_trigger, 208
 - play, 204
 - play_note_off, 212
 - play_note_on, 212
 - pop_redo, 202
 - pop_undo, 202
 - print, 204
 - print_triggers, 204
 - push_trigger_undo, 202
 - push_undo, 202
 - put_event_on_bus, 214
 - remove, 214
 - remove_marked, 212
 - reset_draw_marker, 213
 - reset_draw_trigger_marker, 213
 - select_action_e, 201
 - select_all, 210
 - select_events, 209
 - select_note_events, 209
 - select_trigger, 207
 - selected_trigger_end, 208
 - selected_trigger_start, 208
 - set_beat_width, 202
 - set_beats_per_bar, 202
 - set_dirty, 204
 - set_dirty_mp, 204
 - set_length, 203
 - set_master_midi_bus, 209
 - set_midi_bus, 209
 - set_midi_channel, 204
 - set_orig_tick, 205
 - set_playing, 203
 - set_quantized_rec, 203
 - set_rec_vol, 202
 - set_recording, 203
 - set_snap_tick, 203
 - set_thru, 204
 - set_trigger_offset, 214
 - split_trigger, 205, 214
 - stream_event, 210
 - stretch_selected, 212
 - toggle_queued, 203
 - transpose_notes, 214
 - unpaint_all, 212
 - unselect, 212
 - unselect_triggers, 207

- verify_and_link, 212
 - zero_markers, 212
- seq64::trigger, 215
- seq64::triggers, 216
 - add, 220
 - adjust_offset, 225
 - adjust_offsets_to_length, 221
 - copy, 224
 - get_selected_end, 224
 - get_selected_start, 224
 - get_state, 222
 - grow, 221
 - intersect, 222
 - m_length, 225
 - m_ppqn, 225
 - move_selected, 222
 - next, 225
 - next_trigger, 225
 - operator=, 220
 - paste, 222
 - play, 220
 - remove, 221
 - select, 222
 - set_length, 220
 - split, 221
 - triggers, 219
 - unselect, 222
- seq64::user_instrument, 226
 - controller_active, 227
 - controller_max, 227
 - controller_name, 227
 - copy_definitions, 227
 - m_controller_count, 228
 - m_is_valid, 228
 - set_controller, 227
 - set_defaults, 227
 - set_name, 227
- seq64::user_instrument_t, 228
 - controllers, 228
 - controllers_active, 228
 - instrument, 228
- seq64::user_midi_bus, 228
 - channel_count, 229
 - channel_max, 230
 - copy_definitions, 230
 - instrument, 230
 - m_channel_count, 230
 - m_is_valid, 230
 - set_defaults, 229
 - set_instrument, 230
- seq64::user_midi_bus_t, 230
 - alias, 231
 - instrument, 231
- seq64::user_settings, 231
 - bus, 238
 - Busses, 237
 - control_height, 239
 - controller_active, 238
 - controller_name, 238
 - dump_summary, 239
 - get_globals, 238
 - grid_style_black, 237
 - grid_style_max, 237
 - grid_style_normal, 237
 - grid_style_white, 237
 - instrument, 238
 - Instruments, 237
 - m_control_height, 241
 - m_global_seq_feature_save, 241
 - m_gmute_tracks, 243
 - m_grid_brackets, 240
 - m_grid_style, 240
 - m_instruments, 240
 - m_mainwid_border, 240
 - m_mainwid_x, 243
 - m_mainwnd_cols, 240
 - m_mainwnd_rows, 240
 - m_max_sequence, 243
 - m_max_sets, 240
 - m_midi_beat_width, 242
 - m_midi_beats_per_measure, 242
 - m_midi_beats_per_minute, 242
 - m_midi_buses, 240
 - m_midi_buss_override, 242
 - m_midi_ppqn, 242
 - m_save_user_config, 243
 - m_seqarea_seq_x, 243
 - m_seqarea_x, 243
 - m_seqchars_x, 242
 - m_seqedit_bgsequence, 241
 - m_seqedit_key, 241
 - m_seqedit_scale, 241
 - m_seqs_in_set, 242
 - m_text_x, 241
 - m_use_new_font, 241
 - mainwid_border, 239
 - mainwid_grid_style_t, 237
 - mainwid_spacing, 239
 - mainwnd_cols, 238
 - mainwnd_rows, 238
 - max_sets, 238
 - mc_baseline_ppqn, 244
 - mc_max_zoom, 243
 - mc_min_zoom, 243
 - midi_beat_width, 239
 - midi_beats_per_bar, 239
 - midi_beats_per_minute, 239
 - midi_buss_override, 239
 - midi_ppqn, 239
 - private_bus, 240
 - private_instrument, 240
 - set_defaults, 238
 - set_globals, 238
 - text_x, 239
 - text_y, 239
 - user_settings, 238

- zoom, [238](#)
- seq64::userfile, [244](#)
 - dump_setting_summary, [245](#)
 - parse, [245](#)
 - write, [245](#)
- seq_clear_perf
 - seq64::seqmenu, [186](#)
- seq_copy
 - seq64::seqmenu, [186](#)
- seq_cut
 - seq64::seqmenu, [186](#)
- seq_edit
 - seq64::seqmenu, [186](#)
- seq_from_xy
 - seq64::mainwid, [81](#)
- seq_in_playing_screen
 - seq64::perform, [140](#)
- seq_paste
 - seq64::seqmenu, [186](#)
- seqdata
 - seq64::seqdata, [165](#)
- seqedit
 - seq64::seqedit, [170](#)
- seqmenu
 - seq64::seqmenu, [186](#)
- sequence_count
 - seq64::perform, [131](#)
- sequence_label
 - seq64::perform, [139](#)
- sequence_playing_off
 - seq64::perform, [138](#)
- sequence_playing_on
 - seq64::perform, [138](#)
- set
 - seq64::keybindentry, [63](#)
- set_active
 - seq64::perform, [135](#)
- set_adding
 - seq64::Seq24PerfInput, [160](#)
 - seq64::Seq24SeqEventInput, [161](#)
 - seq64::Seq24SeqRollInput, [162](#)
- set_all_key_events
 - seq64::keys_perform, [66](#)
 - seq64::keys_perform_gtk2, [69](#)
- set_all_key_groups
 - seq64::keys_perform, [66](#)
 - seq64::keys_perform_gtk2, [69](#)
- set_alsa_client_id
 - seq64::lash, [73](#)
- set_background_sequence
 - seq64::seqedit, [172](#)
 - seq64::seqroll, [191](#)
- set_beat_width
 - seq64::perfedit, [119](#)
 - seq64::sequence, [202](#)
- set_beats_per_bar
 - seq64::perfedit, [119](#)
 - seq64::sequence, [202](#)
- set_beats_per_minute
 - seq64::perform, [137](#)
- set_controller
 - seq64::user_instrument, [227](#)
- set_data
 - seq64::event, [36](#)
- set_data_type
 - seq64::seqdata, [165](#)
 - seq64::seqedit, [172](#)
 - seq64::seqevent, [177](#)
 - seq64::seqroll, [191](#)
- set_defaults
 - seq64::user_instrument, [227](#)
 - seq64::user_midi_bus, [229](#)
 - seq64::user_settings, [238](#)
- set_dirty
 - seq64::sequence, [204](#)
- set_dirty_mp
 - seq64::sequence, [204](#)
- set_globals
 - seq64::user_settings, [238](#)
- set_group_mute_state
 - seq64::perform, [138](#)
- set_guides
 - seq64::perfedit, [119](#)
 - seq64::perfroll, [149](#)
 - seq64::perftime, [155](#)
- set_hint_state
 - seq64::seqkeys, [182](#)
- set_input
 - seq64::midibus, [100](#)
- set_input_bus
 - seq64::perform, [140](#)
- set_instrument
 - seq64::user_midi_bus, [230](#)
- set_key
 - seq64::seqedit, [171](#)
- set_key_event
 - seq64::keys_perform, [66](#)
 - seq64::perform, [143](#)
- set_key_group
 - seq64::keys_perform, [67](#)
 - seq64::perform, [143](#)
- set_keys
 - seq64::keys_perform, [65](#)
- set_left_tick
 - seq64::perform, [132](#)
- set_length
 - seq64::sequence, [203](#)
 - seq64::triggers, [220](#)
- set_line
 - seq64::gui_drawingarea_gtk2, [51](#)
- set_master_midi_bus
 - seq64::sequence, [209](#)
- set_measures
 - seq64::seqedit, [171](#)
- set_midi_bus
 - seq64::seqedit, [171](#)

- seq64::sequence, 209
- set_midi_channel
 - seq64::seqedit, 171
 - seq64::sequence, 204
- set_name
 - seq64::user_instrument, 227
- set_note_length
 - seq64::seqedit, 170
- set_offset
 - seq64::perform, 139
- set_orig_tick
 - seq64::sequence, 205
- set_orig_ticks
 - seq64::perform, 137
- set_playing
 - seq64::sequence, 203
- set_playing_screensets
 - seq64::perform, 135
- set_ppqn
 - seq64::perfroll, 150
- set_quantized_rec
 - seq64::sequence, 203
- set_rec_vol
 - seq64::sequence, 202
- set_recording
 - seq64::sequence, 203
- set_right_tick
 - seq64::perform, 133
- set_scale
 - seq64::seqedit, 171
- set_screen_set_notepad
 - seq64::perform, 134
- set_screensets
 - seq64::mainwid, 80
 - seq64::perform, 134
- set_sequence_control_status
 - seq64::perform, 138
- set_snap
 - seq64::seqedit, 170
 - seq64::seqevent, 177
- set_snap_tick
 - seq64::sequence, 203
- set_status
 - seq64::event, 36
- set_thru
 - seq64::sequence, 204
- set_trigger_offset
 - seq64::sequence, 214
- set_was_active
 - seq64::perform, 136
- set_zoom
 - seq64::seqdata, 165
 - seq64::seqedit, 170
- show_ui_sequence_key
 - seq64::keys_perform, 66
 - seq64::perform, 139
- show_ui_sequence_number
 - seq64::keys_perform, 66
- seq64::perform, 139
- signal_action
 - seq64::mainwnd, 90
- sm_internal_keys
 - seq64::gui_assistant_gtk2, 48
- sm_mc_dummy
 - seq64::perform, 144
- smf0_increment
 - seq64::midifile, 106
- snap_x
 - seq64::perfroll, 150
 - seq64::seqevent, 178
 - seq64::seqroll, 192
- split
 - seq64::triggers, 221
- split_trigger
 - seq64::sequence, 205, 214
- start
 - seq64::perform, 135
- start_paste
 - seq64::seqevent, 178
- start_playing
 - seq64::perfeddit, 120
 - seq64::perform, 139
 - seq64::perfroll, 150
- stop
 - seq64::jack_assistant, 60
 - seq64::perform, 135
- stop_playing
 - seq64::perfeddit, 120
 - seq64::perfroll, 151
- stream_event
 - seq64::sequence, 210
- stretch_selected
 - seq64::sequence, 212
- sysex
 - seq64::midibus, 100
- text_x
 - seq64::user_settings, 239
- text_y
 - seq64::user_settings, 239
- timeout
 - seq64::mainwid, 82
 - seq64::perfeddit, 120
- timer_callback
 - seq64::mainwnd, 89
- toggle_queued
 - seq64::sequence, 203
- transpose_notes
 - seq64::sequence, 214
- triggers
 - seq64::triggers, 219
- type
 - seq64::keybindentry, 63
- undo
 - seq64::perfeddit, 120
- unpaint_all

- seq64::sequence, 212
- unselect
 - seq64::sequence, 212
 - seq64::triggers, 222
- unselect_triggers
 - seq64::sequence, 207
- unset_mode_group_learn
 - seq64::perform, 135
- unset_sequence_control_status
 - seq64::perform, 138
- update_background
 - seq64::seqroll, 191
- update_markers
 - seq64::mainwid, 80
- update_sequence_on_window
 - seq64::mainwid, 80
- update_sizes
 - seq64::perfroll, 150
 - seq64::seqdata, 165
 - seq64::sequevent, 177
 - seq64::seqroll, 191
- update_window_title
 - seq64::mainwnd, 89
- user_settings
 - seq64::user_settings, 238
- valid_sequence
 - seq64::mainwid, 81
- varinum_size
 - seq64::midifile, 110
- verify_and_link
 - seq64::event_list, 42
 - seq64::sequence, 212
- WHITE
 - seq64::font, 44
- write
 - seq64::midifile, 104
 - seq64::optionsfile, 116
 - seq64::userfile, 245
- write_byte
 - seq64::midifile, 108
- write_long
 - seq64::midifile, 108
- write_prop_header
 - seq64::midifile, 109
- write_proprietary_track
 - seq64::midifile, 110
- write_seq_number
 - seq64::midifile, 109
- write_short
 - seq64::midifile, 108
- write_track_name
 - seq64::midifile, 109
- write_varinum
 - seq64::midifile, 108
- x_to_w
 - seq64::sequevent, 178
- xy_to_rect
 - seq64::seqdata, 165
- YELLOW_ON_BLACK
 - seq64::font, 44
- zero_markers
 - seq64::sequence, 212
- zoom
 - seq64::user_settings, 238