

Sequencer64 Developer Reference Manual

0.9.10.1

Generated by Doxygen 1.8.11

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
2.2.3	Meta Events Summary	6
2.2.3.1	Sequence Number (0x00)	6
2.2.3.2	Track/Sequence Name (0x03)	7
2.2.3.3	End of Track (0x2F)	7
2.2.3.4	Set Tempo Event (0x51)	7
2.2.3.5	Time Signature Event (0x58)	8
2.2.3.6	SysEx Event (0xF0)	9
2.2.3.7	Sequencer Specific (0x7F)	10
2.2.3.8	Non-Specific End of Sequence	10
2.3	SMF 0 Parsing	10
2.4	Running Status	11

3 JACK, Live, and Song Modes in Sequencer64	13
3.1 Introduction	13
3.2 JACK Functions	13
3.2.1 jack_client_open()	14
3.2.2 jack_on_shutdown()	14
3.2.3 jack_set_sync_callback()	14
3.2.4 jack_set_process_callback()	15
3.2.5 jack_set_session_callback()	15
3.2.6 jack_activate()	15
3.2.7 jack_release_timebase()	15
3.2.8 jack_client_close()	15
3.2.9 jack_transport_start()	15
3.2.10 jack_transport_stop()	15
3.2.11 jack_transport_locate()	15
3.2.12 jack_transport_reposition()	16
3.2.13 jack_transport_query()	16
3.3 Modes Operation	16
3.3.1 No JACK, Live Mode	16
3.3.2 No JACK, Song Mode	16
3.3.3 JACK Transport	17
3.4 Breakage	17
3.5 JACK References	18
4 User Testing of Sequencer64 with Yoshimi	19
4.1 Introduction	19
4.2 Smoke Test	19
4.3 Tests in the Patterns Window	20
4.3.1 Button Clicks on a Pattern	21
4.3.2 Patterns Window Key Shortcuts	21
4.3.3 The Sequencer64 User File	21
4.4 Tests Using Valgrind	21
4.4.1 Valgrind Suppressions	22
4.4.2 Full Valgrind Leak-Checking	22
4.4.2.1 Leak-Checking Basic Operation	23
4.5 Specific Fault Debugging	23
4.6 Snipping of a MIDI file.	23

5	Licenses	25
5.1	License Terms for the This Project.	25
5.2	XPC Application License	25
5.3	XPC Library License	26
5.4	XPC Documentation License	26
5.5	XPC Affero License	27
5.6	XPC License Summary	27
6	Todo List	29
7	Deprecated List	31
8	Namespace Index	33
8.1	Namespace List	33
9	Hierarchical Index	35
9.1	Class Hierarchy	35
10	Data Structure Index	37
10.1	Data Structures	37
11	Namespace Documentation	41
11.1	seq64 Namespace Reference	41
11.1.1	Detailed Description	48
11.1.2	Typedef Documentation	48
11.1.2.1	midipulse	48
11.1.3	Enumeration Type Documentation	48
11.1.3.1	seq_modifier_t	48
11.1.3.2	seq_event_type_t	48
11.1.3.3	seq_scroll_direction_t	49
11.1.3.4	clock_e	49
11.1.3.5	interaction_method_t	49
11.1.3.6	c_music_scales	49
11.1.3.7	draw_type	49

11.1.4 Function Documentation	50
11.1.4.1 extract_timing_numbers(const std::string &s, std::string &part_1, std::string &part_2, std::string &part_3, std::string &fraction)	50
11.1.4.2 pulses_to_string(midipulse p)	50
11.1.4.3 pulses_to_measurestring(midipulse p, const midi_timing &seqparms)	50
11.1.4.4 pulses_to_midi_measures(midipulse p, const midi_timing &seqparms, midi_↔ measures &measures)	50
11.1.4.5 pulses_to_timestring(midipulse p, int bpm, int ppqn)	51
11.1.4.6 pulses_to_timestring(midipulse p, const midi_timing &timinginfo)	51
11.1.4.7 measurestring_to_pulses(const std::string &measures, const midi_timing &seqparms)	51
11.1.4.8 midi_measures_to_pulses(const midi_measures &measures, const midi_timing &seqparms)	52
11.1.4.9 timestring_to_pulses(const std::string ×tring, int bpm, int ppqn)	52
11.1.4.10 string_to_pulses(const std::string &s, const midi_timing &mt)	52
11.1.4.11 string_to_midibyte(const std::string &s)	53
11.1.4.12 shorten_file_spec(const std::string &fpath, int leng)	53
11.1.4.13 string_not_void(const std::string &s)	53
11.1.4.14 string_is_void(const std::string &s)	54
11.1.4.15 strings_match(const std::string &target, const std::string &x)	54
11.1.4.16 log2_time_sig_value(int tsd)	54
11.1.4.17 tempo_to_bytes(midibyte t[3], int tempo_us)	55
11.1.4.18 beats_per_minute_from_tempo(double tempo)	55
11.1.4.19 tempo_from_beats_per_minute(double bpm)	55
11.1.4.20 pulse_length_us(int bpm, int ppqn)	55
11.1.4.21 delta_time_us_to_ticks(unsigned long us, int bpm, int ppqn)	56
11.1.4.22 ticks_to_delta_time_us(midipulse delta_ticks, int bpm, int ppqn)	56
11.1.4.23 clock_tick_duration_bogus(int bpm, int ppqn)	57
11.1.4.24 clock_ticks_from_ppqn(int ppqn)	57
11.1.4.25 double_ticks_from_ppqn(int ppqn)	57
11.1.4.26 measures_to_ticks(int bpm, int ppqn, int bw, int measures=1)	58
11.1.4.27 help_check(int argc, char *argv[])	58

11.1.4.28	<code>parse_options_files(perform &p, int argc, char *argv[])</code>	59
11.1.4.29	<code>parse_command_line_options(int argc, char *argv[])</code>	59
11.1.4.30	<code>write_options_files(const perform &p)</code>	60
11.1.4.31	<code>to_string(const event &ev)</code>	60
11.1.4.32	<code>file_exists(const std::string &filename)</code>	60
11.1.4.33	<code>file_readable(const std::string &filename)</code>	60
11.1.4.34	<code>file_writable(const std::string &filename)</code>	60
11.1.4.35	<code>file_accessible(const std::string &filename)</code>	61
11.1.4.36	<code>file_executable(const std::string &filename)</code>	61
11.1.4.37	<code>file_is_directory(const std::string &filename)</code>	61
11.1.4.38	<code>make_directory(const std::string &pathname)</code>	61
11.1.4.39	<code>choose_ppqn(int ppqn)</code>	62
11.1.4.40	<code>ppqn_is_valid(int ppqn)</code>	62
11.1.4.41	<code>jack_sync_callback(jack_transport_state_t state, jack_position_t *pos, void *arg)</code>	62
11.1.4.42	<code>jack_shutdown_callback(void *arg)</code>	63
11.1.4.43	<code>jack_timebase_callback(jack_transport_state_t state, jack_nframes_t nframes, jack_position_t *pos, int new_pos, void *arg)</code>	63
11.1.4.44	<code>jack_session_callback(jack_session_event_t *ev, void *arg)</code>	64
11.1.4.45	<code>create_lash_driver(perform &p, int argc, char **argv)</code>	64
11.1.4.46	<code>lash_driver()</code>	64
11.1.4.47	<code>delete_lash_driver()</code>	64
11.1.4.48	<code>output_thread_func(void *p)</code>	64
11.1.4.49	<code>min(long a, long b)</code>	65
11.1.4.50	<code>font_render()</code>	65
11.1.4.51	<code>adjustment_dummy()</code>	65
11.1.5	Variable Documentation	65
11.1.5.1	<code>c_controller_names</code>	65
11.1.5.2	<code>EVENT_NOTE_OFF</code>	65
11.1.5.3	<code>EVENT_MIDI_SYSEX</code>	66
11.1.5.4	<code>EVENT_NULL_CHANNEL</code>	66
11.1.5.5	<code>c_midibus_output_size</code>	66
11.1.5.6	<code>c_midibus</code>	66
11.1.5.7	<code>c_midi_track_ctrl</code>	67
11.1.5.8	<code>c_scales_policy</code>	67
11.1.5.9	<code>c_scales_transpose_up</code>	68
11.1.5.10	<code>c_scales_transpose_dn_neg</code>	68
11.1.5.11	<code>c_chord_text</code>	68
11.1.5.12	<code>c_max_instruments</code>	68
11.1.5.13	<code>versiontext</code>	69
11.1.5.14	<code>long_options</code>	69
11.1.5.15	<code>s_global_lash_driver</code>	69
11.1.5.16	<code>c_mainwid_x</code>	69
11.1.5.17	<code>c_select_all_notes</code>	69

12 Data Structure Documentation	71
12.1 seq64::AbstractPerfInput Class Reference	71
12.2 seq64::automutex Class Reference	72
12.2.1 Detailed Description	72
12.3 seq64::click Class Reference	72
12.3.1 Detailed Description	73
12.3.2 Constructor & Destructor Documentation	73
12.3.2.1 click()	73
12.3.2.2 click(int x, int y, int button=SEQ64_CLICK_BUTTON_LEFT, bool press=true, seq_modifier_t modkey=SEQ64_NO_MASK)	73
12.3.2.3 click(const click &rhs)	74
12.3.3 Member Function Documentation	74
12.3.3.1 operator=(const click &rhs)	74
12.3.4 Field Documentation	74
12.3.4.1 m_x	74
12.3.4.2 m_y	74
12.3.4.3 m_button	74
12.3.4.4 m_modifier	74
12.4 seq64::condition_var Class Reference	75
12.4.1 Detailed Description	76
12.4.2 Field Documentation	76
12.4.2.1 cond	76
12.5 seq64::configfile Class Reference	76
12.5.1 Constructor & Destructor Documentation	77
12.5.1.1 configfile(const std::string &name)	77
12.5.2 Member Function Documentation	77
12.5.2.1 next_data_line(std::ifstream &file)	77
12.5.2.2 line_after(std::ifstream &file, const std::string &tag)	78
12.5.3 Field Documentation	78
12.5.3.1 m_line	78
12.6 seq64::editable_event Class Reference	78

12.6.1	Detailed Description	82
12.6.2	Member Enumeration Documentation	82
12.6.2.1	category_t	82
12.6.2.2	timestamp_format_t	82
12.6.3	Constructor & Destructor Documentation	83
12.6.3.1	editable_event(const editable_events &parent)	83
12.6.3.2	editable_event(const editable_events &parent, const event &ev)	83
12.6.3.3	editable_event(const editable_event &rhs)	83
12.6.4	Member Function Documentation	83
12.6.4.1	value_to_name(midibyte value, category_t cat)	83
12.6.4.2	name_to_value(const std::string &name, category_t cat)	83
12.6.4.3	category(category_t c)	84
12.6.4.4	category(const std::string &cs)	84
12.6.4.5	timestamp(midipulse ts)	84
12.6.4.6	timestamp(const std::string &ts_string)	84
12.6.4.7	time_as_measures()	84
12.6.4.8	time_as_minutes()	85
12.6.4.9	set_status_from_string(const std::string &ts, const std::string &s, const std::string &sd0, const std::string &sd1)	85
12.6.4.10	format_timestamp()	85
12.6.4.11	stock_event_string()	85
12.6.4.12	analyze()	85
12.6.5	Field Documentation	86
12.6.5.1	sm_category_names	86
12.6.5.2	sm_channel_event_names	86
12.6.5.3	sm_system_event_names	86
12.6.5.4	sm_meta_event_names	86
12.6.5.5	sm_prop_event_names	86
12.6.5.6	sm_category_arrays	86
12.6.5.7	m_parent	87
12.6.5.8	m_category	87

12.6.5.9	<code>m_format_timestamp</code>	87
12.6.5.10	<code>m_name_status</code>	87
12.6.5.11	<code>m_name_meta</code>	87
12.7	<code>seq64::editable_events</code> Class Reference	87
12.7.1	Member Typedef Documentation	89
12.7.1.1	<code>Key</code>	89
12.7.2	Constructor & Destructor Documentation	89
12.7.2.1	<code>editable_events(sequence &seq, int bpm)</code>	89
12.7.2.2	<code>editable_events(const editable_events &rhs)</code>	89
12.7.3	Member Function Documentation	89
12.7.3.1	<code>operator=(const editable_events &rhs)</code>	89
12.7.3.2	<code>load_events()</code>	89
12.7.3.3	<code>save_events()</code>	90
12.7.3.4	<code>count() const</code>	90
12.7.3.5	<code>add(const event &e)</code>	90
12.7.3.6	<code>add(const editable_event &e)</code>	90
12.7.4	Field Documentation	91
12.7.4.1	<code>m_current_event</code>	91
12.7.4.2	<code>m_sequence</code>	91
12.7.4.3	<code>m_midi_parameters</code>	91
12.8	<code>seq64::event</code> Class Reference	91
12.8.1	Detailed Description	95
12.8.2	Constructor & Destructor Documentation	96
12.8.2.1	<code>event(const event &rhs)</code>	96
12.8.2.2	<code>~event()</code>	96
12.8.3	Member Function Documentation	96
12.8.3.1	<code>operator=(const event &rhs)</code>	96
12.8.3.2	<code>operator<(const event &rhsevent) const</code>	96
12.8.3.3	<code>check_channel(int channel) const</code>	97
12.8.3.4	<code>is_channel_msg(midibyte m)</code>	97

12.8.3.5	<code>is_one_byte_msg(midibyte m)</code>	98
12.8.3.6	<code>is_two_byte_msg(midibyte m)</code>	98
12.8.3.7	<code>is_note_msg(midibyte m)</code>	98
12.8.3.8	<code>is_desired_cc_or_not_cc(midibyte m, midibyte cc, midibyte datum)</code>	98
12.8.3.9	<code>mod_timestamp(midipulse a_mod)</code>	99
12.8.3.10	<code>set_status(midibyte status)</code>	99
12.8.3.11	<code>set_status(midibyte eventcode, midibyte channel)</code>	99
12.8.3.12	<code>set_channel(midibyte channel)</code>	100
12.8.3.13	<code>set_data(midibyte d1)</code>	100
12.8.3.14	<code>set_data(midibyte d1, midibyte d2)</code>	100
12.8.3.15	<code>get_data(midibyte &d0, midibyte &d1) const</code>	100
12.8.3.16	<code>append_sysex(midibyte *data, int len)</code>	100
12.8.3.17	<code>get_rank()</code> const	101
12.8.4	Field Documentation	101
12.8.4.1	<code>m_status</code>	101
12.8.4.2	<code>m_channel</code>	101
12.8.4.3	<code>m_data</code>	101
12.8.4.4	<code>m_sysex</code>	101
12.8.4.5	<code>m_has_link</code>	101
12.9	<code>seq64::event_list::event_key</code> Class Reference	101
12.9.1	Detailed Description	102
12.9.2	Constructor & Destructor Documentation	102
12.9.2.1	<code>event_key(midipulse tstamp, int rank)</code>	102
12.9.2.2	<code>event_key(const event &e)</code>	102
12.9.3	Member Function Documentation	102
12.9.3.1	<code>operator<(const event_key &rhs) const</code>	102
12.9.4	Field Documentation	103
12.9.4.1	<code>m_timestamp</code>	103
12.9.4.2	<code>m_rank</code>	103
12.10	<code>seq64::event_list</code> Class Reference	103

12.10.1 Detailed Description	105
12.10.2 Constructor & Destructor Documentation	105
12.10.2.1 event_list(const event_list &a_rhs)	105
12.10.3 Member Function Documentation	105
12.10.3.1 operator=(const event_list &a_rhs)	105
12.10.3.2 count() const	105
12.10.3.3 add(const event &e, bool postsort=true)	106
12.10.3.4 unmodify()	106
12.10.3.5 remove(iterator ie)	106
12.10.3.6 clear()	106
12.10.3.7 merge(event_list &el, bool presort=true)	106
12.10.3.8 link_new()	107
12.10.3.9 verify_and_link(midipulse slength)	107
12.10.3.10 mark_out_of_range(midipulse slength)	107
12.10.3.11 mark_all()	108
12.10.3.12 any_selected_notes() const	108
12.10.3.13 count_selected_events(midibyte status, midibyte cc) const	108
12.10.4 Field Documentation	108
12.10.4.1 m_is_modified	108
12.11 seq64::eventedit Class Reference	108
12.11.1 Constructor & Destructor Documentation	111
12.11.1.1 eventedit(perform &p, sequence &seq)	111
12.11.1.2 ~eventedit()	112
12.11.2 Member Function Documentation	112
12.11.2.1 set_seq_title(const std::string &title)	112
12.11.2.2 set_seq_time_sig(const std::string &sig)	112
12.11.2.3 set_seq_ppqn(const std::string &p)	112
12.11.2.4 set_event_category(const std::string &c)	113
12.11.2.5 set_event_timestamp(const std::string &ts)	113
12.11.2.6 set_event_name(const std::string &n)	113

12.11.2.7 set_event_data_0(const std::string &d)	113
12.11.2.8 set_event_data_1(const std::string &d)	113
12.11.2.9 set_dirty(bool flag=true)	113
12.11.2.10v_adjustment(int value)	113
12.11.2.11v_adjustment(int value, int lower, int upper)	114
12.11.2.12handle_insert()	114
12.11.2.13handle_modify()	114
12.11.2.14handle_save()	114
12.11.2.15on_key_press_event(GdkEventKey *ev)	114
12.11.2.16on_delete_event(GdkEventAny *event)	115
12.12seq64::eventslots Class Reference	115
12.12.1 Member Function Documentation	120
12.12.1.1 load_events()	120
12.12.1.2 set_current_event(const editable_events::iterator ei, int index, bool full_redraw=true)	120
12.12.1.3 insert_event(const editable_event &edev)	120
12.12.1.4 insert_event(const std::string &evtimestamp, const std::string &evname, const std::string &evdata0, const std::string &evdata1)	121
12.12.1.5 delete_current_event()	121
12.12.1.6 modify_current_event(const std::string &evtimestamp, const std::string &evname, const std::string &evdata0, const std::string &evdata1)	122
12.12.1.7 save_events()	122
12.12.1.8 select_event(int event_index=SEQ64_NULL_EVENT_INDEX, bool full_redraw=true)	123
12.12.1.9 set_text(const std::string &evcategory, const std::string &evtimestamp, const std::string &evname, const std::string &evdata0, const std::string &evdata1)	123
12.12.1.10convert_y(int y)	123
12.12.1.11draw_event(editable_events::iterator ei, int index)	124
12.12.1.12draw_events()	124
12.12.1.13change_vert()	124
12.12.1.14page_movement(int new_value)	124
12.12.1.15page_topper(editable_events::iterator newcurrent)	125
12.12.1.16decrement_top()	125

12.12.1.17	increment_top()	125
12.12.1.18	decrement_current()	125
12.12.1.19	increment_current()	125
12.12.1.20	decrement_bottom()	126
12.12.1.21	increment_bottom()	126
12.12.1.22	on_realize()	126
12.12.1.23	on_expose_event(GdkEventExpose *ev)	126
12.12.1.24	on_focus_in_event(GdkEventFocus *ev)	126
12.12.1.25	on_size_allocate(Gtk::Allocation &)	126
12.12.1.26	on_move_up()	126
12.12.1.27	on_move_down()	126
12.12.2	Field Documentation	127
12.12.2.1	m_slots_chars	127
12.12.2.2	m_char_w	127
12.12.2.3	m_setbox_w	127
12.12.2.4	m_slots_y	127
12.12.2.5	m_xy_offset	127
12.12.2.6	m_top_index	127
12.12.2.7	m_current_index	127
12.13	seq64::font Class Reference	127
12.13.1	Member Enumeration Documentation	129
12.13.1.1	Color	129
12.13.2	Member Function Documentation	129
12.13.2.1	init(Glib::RefPtr< Gdk::Window > windo)	129
12.13.2.2	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	129
12.13.3	Field Documentation	129
12.13.3.1	m_font_w	130
12.13.3.2	m_font_h	130
12.13.3.3	m_pixmap	130
12.13.3.4	m_black_pixmap	130

12.13.3.5 <code>m_white_pixmap</code>	130
12.13.3.6 <code>m_b_on_y_pixmap</code>	130
12.13.3.7 <code>m_y_on_b_pixmap</code>	130
12.13.3.8 <code>m_b_on_c_pixmap</code>	130
12.13.3.9 <code>m_c_on_b_pixmap</code>	130
12.13.3.10 <code>m_clip_mask</code>	131
12.14 <code>seq64::gui_assistant</code> Class Reference	131
12.14.1 Detailed Description	132
12.14.2 Constructor & Destructor Documentation	132
12.14.2.1 <code>gui_assistant(keys_perform &kp)</code>	132
12.15 <code>seq64::gui_assistant_gtk2</code> Class Reference	132
12.15.1 Member Function Documentation	133
12.15.1.1 <code>lash_timeout_connect(lash *lashobject)</code>	133
12.15.1.2 <code>jack_idle_connect(jack_assistant &jack)</code>	134
12.15.2 Field Documentation	134
12.15.2.1 <code>sm_internal_keys</code>	134
12.16 <code>seq64::gui_drawingarea_gtk2</code> Class Reference	134
12.16.1 Detailed Description	137
12.16.2 Member Function Documentation	137
12.16.2.1 <code>clear_window()</code>	137
12.16.2.2 <code>set_line(Gdk::LineStyle ls, int width=1)</code>	137
12.16.2.3 <code>draw_line(int x1, int y1, int x2, int y2)</code>	137
12.16.2.4 <code>draw_line(const Color &c, int x1, int y1, int x2, int y2)</code>	137
12.16.2.5 <code>draw_line_on_pixmap(int x1, int y1, int x2, int y2)</code>	138
12.16.2.6 <code>draw_line_on_pixmap(const Color &c, int x1, int y1, int x2, int y2)</code>	138
12.16.2.7 <code>draw_line(Glib::RefPtr< Gdk::Pixmap > &pixmap, int x1, int y1, int x2, int y2)</code>	138
12.16.2.8 <code>draw_line(Glib::RefPtr< Gdk::Pixmap > &pixmap, const Color &c, int x1, int y1, int x2, int y2)</code>	138
12.16.2.9 <code>draw_line(Glib::RefPtr< Gdk::Drawable > &drawable, int x1, int y1, int x2, int y2)</code>	139
12.16.2.10 <code>draw_line(Glib::RefPtr< Gdk::Drawable > &drawable, const Color &c, int x1, int y1, int x2, int y2)</code>	139

12.16.2.11	<code>render_string(int x, int y, const std::string &s, font::Color color)</code>	139
12.16.2.12	<code>render_string_on_pixmap(int x, int y, const std::string &s, font::Color color)</code>	139
12.16.2.13	<code>draw_rectangle(int x, int y, int lx, int ly, bool fill=true)</code>	140
12.16.2.14	<code>draw_rectangle(const Color &c, int x, int y, int lx, int ly, bool fill=true)</code>	140
12.16.2.15	<code>draw_rectangle(Glib::RefPtr< Gdk::Drawable > &drawable, int x, int y, int lx, int ly, bool fill=true)</code>	140
12.16.2.16	<code>draw_rectangle(Glib::RefPtr< Gdk::Drawable > &drawable, const Color &c, int x, int y, int lx, int ly, bool fill=true)</code>	141
12.16.2.17	<code>draw_rectangle(Glib::RefPtr< Gdk::Pixmap > &pixmap, int x, int y, int lx, int ly, bool fill=true)</code>	141
12.16.2.18	<code>draw_rectangle(Glib::RefPtr< Gdk::Pixmap > &pixmap, const Color &c, int x, int y, int lx, int ly, bool fill=true)</code>	141
12.16.2.19	<code>draw_rectangle_on_pixmap(int x, int y, int lx, int ly, bool fill=true)</code>	142
12.16.2.20	<code>draw_rectangle_on_pixmap(const Color &c, int x, int y, int lx, int ly, bool fill=true)</code>	142
12.16.2.21	<code>draw_normal_rectangle_on_pixmap(int x, int y, int lx, int ly, bool fill=true)</code>	142
12.16.2.22	<code>scroll_adjust(Gtk::Adjustment &adjust, double step)</code>	142
12.16.2.23	<code>on_realize()</code>	143
12.16.3	Field Documentation	143
12.16.3.1	<code>m_window</code>	143
12.16.3.2	<code>m_pixmap</code>	143
12.16.3.3	<code>m_background</code>	143
12.16.3.4	<code>m_foreground</code>	143
12.16.3.5	<code>m_mainperf</code>	143
12.16.3.6	<code>m_window_x</code>	143
12.16.3.7	<code>m_drop_x</code>	143
12.17	<code>seq64::gui_palette_gtk2</code> Class Reference	144
12.17.1	Detailed Description	145
12.17.2	Constructor & Destructor Documentation	145
12.17.2.1	<code>gui_palette_gtk2()</code>	145
12.17.3	Member Function Documentation	145
12.17.3.1	<code>line_color() const</code>	145
12.17.3.2	<code>progress_color() const</code>	145

12.18seq64::gui_window_gtk2 Class Reference	146
12.18.1 Constructor & Destructor Documentation	147
12.18.1.1 gui_window_gtk2(perform &p, int window_x=0, int window_y=0)	147
12.18.2 Member Function Documentation	147
12.18.2.1 scroll_adjust(Gtk::Adjustment &adjust, double step)	147
12.18.3 Field Documentation	147
12.18.3.1 m_window_x	147
12.18.3.2 m_redraw_period_ms	148
12.19seq64::jack_assistant Class Reference	148
12.19.1 Constructor & Destructor Documentation	150
12.19.1.1 jack_assistant(perform &parent, int bpmminute=SEQ64_DEFAULT_BPM, int ppqn=SEQ64_USE_DEFAULT_PPQN, int bpm=SEQ64_DEFAULT_BEAT← S_PER_MEASURE, int beatwidth=SEQ64_DEFAULT_BEAT_WIDTH)	150
12.19.1.2 ~jack_assistant()	150
12.19.2 Member Function Documentation	150
12.19.2.1 set_beats_per_minute(int bpmminute)	150
12.19.2.2 init()	150
12.19.2.3 deinit()	151
12.19.2.4 session_event()	151
12.19.2.5 start()	151
12.19.2.6 stop()	151
12.19.2.7 position(bool to_left_tick, bool relocate=false)	151
12.19.2.8 output(jack_scratchpad &pad)	152
12.19.2.9 set_ppqn(int ppqn)	152
12.19.2.10info_message(const std::string &msg)	153
12.19.2.11error_message(const std::string &msg)	153
12.19.2.12client_open(const std::string &clientname)	153
12.19.2.13show_statuses(unsigned bits)	154
12.19.2.14show_position(const jack_position_t &pos) const	154
12.19.2.15sync(jack_transport_state_t state=(jack_transport_state_t)(-1))	155
12.19.2.16set_position(midipulse currenttick)	155

12.19.3 Friends And Related Function Documentation	155
12.19.3.1 jack_shutdown_callback	155
12.19.3.2 jack_sync_callback	156
12.19.3.3 jack_timebase_callback	156
12.19.3.4 jack_session_callback	157
12.19.4 Field Documentation	157
12.19.4.1 sm_status_pairs	157
12.20seq64::jack_scratchpad Class Reference	157
12.20.1 Detailed Description	157
12.21seq64::keybindentry Class Reference	157
12.21.1 Member Enumeration Documentation	158
12.21.1.1 type	158
12.21.2 Constructor & Destructor Documentation	158
12.21.2.1 keybindentry(type t, unsigned int *location_to_write=nullptr, perform *p=nullptr, long s=0)	158
12.21.3 Member Function Documentation	159
12.21.3.1 set(unsigned int val)	159
12.21.3.2 on_key_press_event(GdkEventKey *event)	159
12.21.4 Field Documentation	159
12.21.4.1 m_key	159
12.22seq64::keys_perform Class Reference	159
12.22.1 Detailed Description	161
12.22.2 Constructor & Destructor Documentation	161
12.22.2.1 ~keys_perform()	161
12.22.3 Member Function Documentation	161
12.22.3.1 set_keys(const keys_perform_transfer &kpt)	161
12.22.3.2 get_keys(keys_perform_transfer &kpt)	162
12.22.3.3 show_ui_sequence_key() const	162
12.22.3.4 show_ui_sequence_number() const	162
12.22.3.5 key_name(unsigned int key) const	162
12.22.3.6 set_all_key_events()	162

12.22.3.7 set_all_key_groups()	162
12.22.3.8 set_key_event(unsigned int keycode, long sequence_slot)	163
12.22.3.9 set_key_group(unsigned int keycode, long group_slot)	163
12.22.4 Field Documentation	163
12.22.4.1 m_key_show_ui_sequence_number	163
12.22.4.2 m_key_bpm_up	163
12.23seq64::keys_perform_gtk2 Class Reference	163
12.23.1 Detailed Description	165
12.23.2 Constructor & Destructor Documentation	165
12.23.2.1 ~keys_perform_gtk2()	165
12.23.3 Member Function Documentation	165
12.23.3.1 key_name(unsigned int key) const	165
12.23.3.2 set_all_key_events()	165
12.23.3.3 set_all_key_groups()	165
12.24seq64::keys_perform_transfer Struct Reference	165
12.25seq64::keystroke Class Reference	166
12.25.1 Detailed Description	166
12.25.2 Constructor & Destructor Documentation	166
12.25.2.1 keystroke(unsigned int key, bool press=SEQ64_KEYSTROKE_PRESS, int modkey=int(SEQ64_NO_MASK))	166
12.25.2.2 keystroke(const keystroke &rhs)	167
12.25.3 Member Function Documentation	167
12.25.3.1 operator=(const keystroke &rhs)	167
12.25.3.2 is_letter(int ch=SEQ64_KEYSTROKE_BAD_VALUE) const	167
12.25.4 Field Documentation	167
12.25.4.1 m_is_press	167
12.25.4.2 m_key	168
12.25.4.3 m_modifier	168
12.26seq64::lash Class Reference	168
12.26.1 Detailed Description	168
12.26.2 Constructor & Destructor Documentation	168

12.26.2.1	lash(perform &p, int argc, char **argv)	168
12.26.3	Member Function Documentation	169
12.26.3.1	set_alsa_client_id(int id)	169
12.26.3.2	process_events()	169
12.26.3.3	init()	169
12.26.3.4	handle_event(lash_event_t *conf)	169
12.26.3.5	handle_config(lash_config_t *conf)	169
12.27	seq64::maintime Class Reference	169
12.27.1	Detailed Description	171
12.27.2	Constructor & Destructor Documentation	171
12.27.2.1	maintime(perform &p, int ppqn=SEQ64_USE_DEFAULT_PPQN)	171
12.27.3	Member Function Documentation	172
12.27.3.1	idle_progress(midipulse ticks)	172
12.27.3.2	on_realize()	173
12.27.3.3	on_expose_event(GdkEventExpose *ev)	173
12.27.4	Field Documentation	173
12.27.4.1	m_beat_width	173
12.27.4.2	m_bar_width	173
12.27.4.3	m_box_width	173
12.27.4.4	m_box_height	173
12.27.4.5	m_flash_width	173
12.27.4.6	m_flash_height	174
12.27.4.7	m_tick	174
12.27.4.8	m_ppqn	174
12.28	seq64::mainwid Class Reference	174
12.28.1	Constructor & Destructor Documentation	177
12.28.1.1	mainwid(perform &p)	177
12.28.2	Member Function Documentation	177
12.28.2.1	set_screenset(int ss)	177
12.28.2.2	redraw(int seq)	177

12.28.2.3 draw_marker_on_sequence(int seq, int tick)	178
12.28.2.4 update_sequences_on_window()	178
12.28.2.5 update_markers(int ticks)	178
12.28.2.6 valid_sequence(int seq)	178
12.28.2.7 draw_sequence_on_pixmap(int seq)	178
12.28.2.8 draw_sequences_on_pixmap()	179
12.28.2.9 draw_sequence_pixmap_on_window(int seq)	179
12.28.2.10 seq_from_xy(int x, int y)	179
12.28.2.11 timeout()	179
12.28.2.12 calculate_base_sizes(int seq, int &basex, int &basey)	180
12.28.2.13 on_realize()	180
12.28.2.14 on_expose_event(GdkEventExpose *ev)	180
12.28.2.15 on_button_press_event(GdkEventButton *ev)	180
12.28.2.16 on_button_release_event(GdkEventButton *ev)	181
12.28.2.17 on_motion_notify_event(GdkEventMotion *p0)	181
12.28.2.18 on_focus_in_event(GdkEventFocus *)	181
12.28.2.19 on_focus_out_event(GdkEventFocus *)	182
12.28.3 Field Documentation	182
12.28.3.1 m_moving_seq	182
12.28.3.2 m_button_down	182
12.28.3.3 m_screenset_slots	182
12.28.3.4 m_screenset_offset	182
12.29 seq64::mainwnd Class Reference	182
12.29.1 Constructor & Destructor Documentation	186
12.29.1.1 mainwnd(perform &a_p, bool allowperf2=true, int ppqn=SEQ64_USE_DEFAULT_PPQN)	186
12.29.2 Member Function Documentation	187
12.29.2.1 open_file(const std::string &)	187
12.29.2.2 ppqn(int ppqn)	187
12.29.2.3 file_import_dialog()	187
12.29.2.4 about_dialog()	188

12.29.2.5 adj_callback_ss()	188
12.29.2.6 adj_callback_bpm()	188
12.29.2.7 edit_callback_notepad()	188
12.29.2.8 timer_callback()	188
12.29.2.9 set_image(bool isrunning)	188
12.29.2.10 start_playing()	189
12.29.2.11 pause_playing()	189
12.29.2.12 stop_playing()	189
12.29.2.13 toggle_playing()	189
12.29.2.14 open_performance_edit()	189
12.29.2.15 open_performance_edit_2()	189
12.29.2.16 update_window_title()	189
12.29.2.17 new_file()	190
12.29.2.18 save_file()	190
12.29.2.19 signal_action(Glib::IOCondition condition)	190
12.29.2.20 on_delete_event(GdkEventAny *a_e)	190
12.29.2.21 on_key_press_event(GdkEventKey *a_ev)	190
12.29.2.22 on_key_release_event(GdkEventKey *a_ev)	190
12.29.2.23 on_grouplearnchange(bool state)	190
12.29.3 Field Documentation	191
12.29.3.1 m_sigpipe	191
12.29.3.2 m_ppqn	191
12.29.3.3 m_main_wid	191
12.29.3.4 m_button_play	191
12.29.3.5 m_adjust_load_offset	191
12.30 seq64::mastermidibus Class Reference	191
12.30.1 Constructor & Destructor Documentation	193
12.30.1.1 mastermidibus(int ppqn=SEQ64_USE_DEFAULT_PPQN, int bpm=c_beats_per_minute)	193
12.30.1.2 ~mastermidibus()	194
12.30.2 Member Function Documentation	194

12.30.2.1	init(int ppqn)	194
12.30.2.2	set_beats_per_minute(int bpm)	194
12.30.2.3	set_ppqn(int ppqn)	194
12.30.2.4	get_midi_out_bus_name(int a_bus)	195
12.30.2.5	get_midi_in_bus_name(int a_bus)	195
12.30.2.6	flush()	195
12.30.2.7	start()	195
12.30.2.8	stop()	195
12.30.2.9	clock(midipulse a_tick)	195
12.30.2.10	continue_from(midipulse a_tick)	196
12.30.2.11	init_clock(midipulse a_tick)	196
12.30.2.12	poll_for_midi()	196
12.30.2.13	s_more_input()	196
12.30.2.14	get_midi_event(event *a_in)	196
12.30.2.15	set_sequence_input(bool a_state, sequence *a_seq)	196
12.30.2.16	sysex(event *a_event)	197
12.30.2.17	port_start(int a_client, int a_port)	197
12.30.2.18	port_exit(int a_client, int a_port)	197
12.30.2.19	play(bussbyte bus, event *a_e24, midibyte a_channel)	197
12.30.2.20	set_clock(bussbyte bus, clock_e a_clock_type)	198
12.30.2.21	get_clock(bussbyte bus)	198
12.30.2.22	set_input(bussbyte bus, bool a_inputing)	198
12.30.2.23	get_input(bussbyte bus)	198
12.30.3	Field Documentation	198
12.30.3.1	m_beats_per_minute	198
12.31	seq64::midi_container Class Reference	199
12.31.1	Member Function Documentation	200
12.31.1.1	fill(int tracknumber)	200
12.31.1.2	put(midibyte b)=0	201
12.31.1.3	get()=0	201

12.31.1.4 position() const	201
12.31.1.5 add_variable(midipulse v)	201
12.31.1.6 add_long(midipulse x)	201
12.32seq64::midi_list Class Reference	202
12.32.1 Member Typedef Documentation	203
12.32.1.1 CharList	203
12.32.2 Member Function Documentation	203
12.32.2.1 put(midibyte b)	203
12.32.2.2 get()	203
12.33seq64::midi_measures Class Reference	203
12.33.1 Detailed Description	204
12.33.2 Field Documentation	204
12.33.2.1 m_divisions	204
12.34seq64::midi_splitter Class Reference	204
12.34.1 Detailed Description	205
12.34.2 Constructor & Destructor Documentation	205
12.34.2.1 midi_splitter(int ppqn=SEQ64_USE_DEFAULT_PPQN)	205
12.34.3 Member Function Documentation	206
12.34.3.1 increment(int channel)	206
12.34.3.2 split(perform &p, int screenset)	206
12.34.3.3 ppqn() const	206
12.34.3.4 split_channel(const sequence &main_seq, sequence *seq, int channel)	207
12.34.4 Field Documentation	207
12.34.4.1 m_smf0_channels_count	207
12.34.4.2 m_smf0_channels	207
12.34.4.3 m_smf0_seq_number	207
12.35seq64::midi_timing Class Reference	207
12.35.1 Detailed Description	208
12.35.2 Field Documentation	208
12.35.2.1 m_beats_per_minute	208

12.35.2.2 m_beats_per_measure	209
12.35.2.3 m_beat_width	209
12.35.2.4 m_ppqn	209
12.36seq64::midi_vector Class Reference	209
12.36.1 Member Function Documentation	210
12.36.1.1 put(midibyte b)	210
12.36.1.2 get()	210
12.37seq64::midibus Class Reference	210
12.37.1 Constructor & Destructor Documentation	212
12.37.1.1 midibus(int localclient, int destclient, int destport, snd_seq_t *seq, const char *client_name, const char *port_name, int id, int queue, int ppqn=SEQ64_U↵ SE_DEFAULT_PPQN)	212
12.37.1.2 midibus(int localclient, snd_seq_t *seq, int id, int queue, int ppqn=SEQ64_US↵ E_DEFAULT_PPQN)	213
12.37.2 Member Function Documentation	213
12.37.2.1 init_out()	213
12.37.2.2 init_in()	213
12.37.2.3 deinit_in()	213
12.37.2.4 init_out_sub()	214
12.37.2.5 init_in_sub()	214
12.37.2.6 play(event *e24, midibyte channel)	214
12.37.2.7 sysex(event *e24)	214
12.37.2.8 clock(midipulse tick)	214
12.37.2.9 continue_from(midipulse tick)	214
12.37.2.10init_clock(midipulse tick)	214
12.37.2.11set_input(bool inputing)	215
12.37.3 Field Documentation	215
12.37.3.1 m_clock_mod	215
12.38seq64::midifile Class Reference	215
12.38.1 Detailed Description	217
12.38.2 Constructor & Destructor Documentation	218

12.38.2.1 midifile(const std::string &name, int ppqn=SEQ64_USE_DEFAULT_PPQN, bool oldformat=false, bool globalbgs=true)	218
12.38.3 Member Function Documentation	218
12.38.3.1 parse(perform &a_perf, int a_screen_set=0)	218
12.38.3.2 write(perform &a_perf)	219
12.38.3.3 ppqn() const	219
12.38.3.4 parse_smf_0(perform &p, int screenset)	220
12.38.3.5 parse_smf_1(perform &p, int screenset, bool is_smf0=false)	220
12.38.3.6 parse_prop_header(int file_size)	220
12.38.3.7 parse_proprietary_track(perform &a_perf, int file_size)	221
12.38.3.8 pow2(int logbase2)	222
12.38.3.9 checklen(midilong len, midibyte type)	222
12.38.3.10 add_trigger(sequence &seq, midishort ppqn)	222
12.38.3.11 read_long()	222
12.38.3.12 read_varinum()	222
12.38.3.13 write_long(midilong value)	223
12.38.3.14 write_short(midishort value)	223
12.38.3.15 read_byte_array(midibyte *b, int len)	223
12.38.3.16 write_byte(midibyte c)	223
12.38.3.17 write_varinum(midilong)	223
12.38.3.18 write_track_name(const std::string &trackname)	224
12.38.3.19 read_track_name()	224
12.38.3.20 write_seq_number(midishort seqnum)	224
12.38.3.21 read_seq_number()	224
12.38.3.22 write_prop_header(midilong tag, long len)	224
12.38.3.23 write_proprietary_track(perform &a_perf)	225
12.38.3.24 varinum_size(long len) const	226
12.38.3.25 prop_item_size(long datalen) const	226
12.38.3.26 errdump(const std::string &msg)	226
12.38.3.27 errdump(const std::string &msg, unsigned long p)	226
12.38.3.28 s_sysex_special_id(midibyte ch)	227

12.38.4 Field Documentation	227
12.38.4.1 m_file_size	227
12.38.4.2 m_error_message	227
12.38.4.3 m_error_is_fatal	227
12.38.4.4 m_disable_reported	227
12.38.4.5 m_pos	227
12.38.4.6 m_data	227
12.38.4.7 m_char_list	227
12.38.4.8 m_new_format	228
12.38.4.9 m_smf0_splitter	228
12.39seq64::mutex Class Reference	228
12.39.1 Field Documentation	229
12.39.1.1 sm_recursive_mutex	229
12.40seq64::options Class Reference	229
12.40.1 Member Enumeration Documentation	230
12.40.1.1 button	230
12.40.2 Field Documentation	230
12.40.2.1 m_notebook	230
12.41seq64::optionsfile Class Reference	230
12.41.1 Detailed Description	231
12.41.2 Member Function Documentation	232
12.41.2.1 parse(perform &perf)	232
12.41.2.2 write(const perform &perf)	233
12.42seq64::perfedit Class Reference	233
12.42.1 Detailed Description	236
12.42.2 Constructor & Destructor Documentation	236
12.42.2.1 perfedit(perform &p, bool second_perfedit=false, int ppqn=SEQ64_USE_DEFA↵ ULT_PPQN)	236
12.42.2.2 ~perfedit()	237
12.42.3 Member Function Documentation	237
12.42.3.1 init_before_show()	237

12.42.3.2 enqueue_draw(bool forward=true)	237
12.42.3.3 zoom_check(int z)	237
12.42.3.4 enregister_peer(perfedit *peer)	237
12.42.3.5 set_beats_per_bar(int bpm)	237
12.42.3.6 set_beat_width(int bw)	237
12.42.3.7 set_guides()	238
12.42.3.8 grow()	238
12.42.3.9 expand()	238
12.42.3.10 collapse()	238
12.42.3.11 copy()	238
12.42.3.12 undo()	238
12.42.3.13 timeout()	238
12.42.3.14 start_playing()	238
12.42.3.15 pause_playing()	238
12.42.3.16 toggle_playing()	239
12.42.4 Field Documentation	239
12.42.4.1 m_bpm	239
12.42.4.2 m_bw	239
12.43 seq64::perfnames Class Reference	239
12.43.1 Detailed Description	242
12.43.2 Constructor & Destructor Documentation	242
12.43.2.1 perfnames(perform &p, perfedit &parent, Gtk::Adjustment &vadjust)	242
12.43.3 Member Function Documentation	242
12.43.3.1 enqueue_draw()	242
12.43.3.2 draw_sequence(int sequence)	242
12.43.3.3 on_realize()	242
12.43.3.4 on_expose_event(GdkEventExpose *ev)	242
12.43.3.5 on_size_allocate(Gtk::Allocation &)	242
12.43.4 Field Documentation	242
12.43.4.1 m_parent	242

12.43.4.2 m_names_chars	243
12.43.4.3 m_char_w	243
12.43.4.4 m_setbox_w	243
12.43.4.5 m_namebox_w	243
12.43.4.6 m_names_y	243
12.43.4.7 m_xy_offset	243
12.44seq64::perform Class Reference	243
12.44.1 Detailed Description	251
12.44.2 Constructor & Destructor Documentation	251
12.44.2.1 perform(gui_assistant &mygui, int ppqn=SEQ64_USE_DEFAULT_PPQN)	251
12.44.2.2 ~perform()	252
12.44.3 Member Function Documentation	252
12.44.3.1 modify()	252
12.44.3.2 sequence_count() const	252
12.44.3.3 clear_all()	252
12.44.3.4 launch(int ppqn)	252
12.44.3.5 new_sequence(int seq)	252
12.44.3.6 add_sequence(sequence *seq, int perf)	253
12.44.3.7 delete_sequence(int seq)	253
12.44.3.8 clear_sequence_triggers(int seq)	253
12.44.3.9 finish()	253
12.44.3.10set_left_tick(midipulse tick, bool setstart=true)	254
12.44.3.11get_max_tick() const	254
12.44.3.12set_right_tick(midipulse tick, bool setstart=true)	254
12.44.3.13move_triggers(bool direction)	254
12.44.3.14copy_triggers()	254
12.44.3.15push_trigger_undo()	255
12.44.3.16get_max_trigger()	255
12.44.3.17midi_control_toggle(int seq)	255
12.44.3.18midi_control_on(int seq)	255

12.44.3.19	<code>midi_control_off(int seq)</code>	255
12.44.3.20	<code>handle_midi_control(int control, bool state)</code>	255
12.44.3.21	<code>get_screen_set_notepad(int screen_set) const</code>	255
12.44.3.22	<code>set_screen_set_notepad(int screenset, const std::string &note)</code>	256
12.44.3.23	<code>set_screenset(int ss)</code>	256
12.44.3.24	<code>set_playing_screenset()</code>	256
12.44.3.25	<code>select_group_mute(int g_mute)</code>	256
12.44.3.26	<code>unset_mode_group_learn()</code>	256
12.44.3.27	<code>select_mute_group(int group)</code>	257
12.44.3.28	<code>start(bool state)</code>	257
12.44.3.29	<code>stop()</code>	257
12.44.3.30	<code>position_jack(bool state)</code>	257
12.44.3.31	<code>all_notes_off()</code>	257
12.44.3.32	<code>set_active(int seq, bool active)</code>	257
12.44.3.33	<code>set_was_active(int seq)</code>	258
12.44.3.34	<code>is_dirty_main(int seq)</code>	258
12.44.3.35	<code>is_dirty_edit(int seq)</code>	258
12.44.3.36	<code>is_dirty_perf(int seq)</code>	258
12.44.3.37	<code>is_dirty_names(int seq)</code>	258
12.44.3.38	<code>is_active(int seq) const</code>	259
12.44.3.39	<code>get_sequence(int seq)</code>	259
12.44.3.40	<code>reset_sequences(bool pause=false)</code>	259
12.44.3.41	<code>play(midipulse tick)</code>	259
12.44.3.42	<code>set_orig_ticks(midipulse tick)</code>	260
12.44.3.43	<code>set_beats_per_minute(int bpm)</code>	260
12.44.3.44	<code>set_sequence_control_status(int status)</code>	260
12.44.3.45	<code>unset_sequence_control_status(int status)</code>	260
12.44.3.46	<code>sequence_playing_on(int seq)</code>	260
12.44.3.47	<code>sequence_playing_off(int seq)</code>	260
12.44.3.48	<code>set_group_mute_state(int g_track, bool mute_state)</code>	261

12.44.3.49	<code>get_group_mute_state(int g_track)</code>	261
12.44.3.50	<code>output_func()</code>	261
12.44.3.51	<code>set_offset(int offset)</code>	261
12.44.3.52	<code>save_playing_state()</code>	262
12.44.3.53	<code>show_ui_sequence_key() const</code>	262
12.44.3.54	<code>show_ui_sequence_number() const</code>	262
12.44.3.55	<code>lookup_keyevent_key(long seqnum)</code>	262
12.44.3.56	<code>start_playing(bool songmode=false)</code>	262
12.44.3.57	<code>pause_playing()</code>	263
12.44.3.58	<code>stop_playing()</code>	263
12.44.3.59	<code>start_key(bool songmode=false)</code>	263
12.44.3.60	<code>pause_key(bool songmode=false)</code>	263
12.44.3.61	<code>stop_key()</code>	263
12.44.3.62	<code>decrement_beats_per_minute()</code>	264
12.44.3.63	<code>increment_beats_per_minute()</code>	264
12.44.3.64	<code>highlight(const sequence &seq) const</code>	264
12.44.3.65	<code>sequence_label(const sequence &seq)</code>	264
12.44.3.66	<code>set_input_bus(int bus, bool input_active)</code>	264
12.44.3.67	<code>mainwnd_key_event(const keystroke &k)</code>	265
12.44.3.68	<code>perfroll_key_event(const keystroke &k, int drop_sequence)</code>	265
12.44.3.69	<code>playback_key_event(const keystroke &k, bool songmode=false)</code>	265
12.44.3.70	<code>launch_input_thread()</code>	265
12.44.3.71	<code>launch_output_thread()</code>	266
12.44.3.72	<code>init_jack()</code>	266
12.44.3.73	<code>deinit_jack()</code>	266
12.44.3.74	<code>seq_in_playing_screen(int seq)</code>	266
12.44.3.75	<code>s_midi_control_valid(int seq) const</code>	266
12.44.3.76	<code>s_screenset_valid(int screenset) const</code>	266
12.44.3.77	<code>s_seq_valid(int seq) const</code>	267
12.44.3.78	<code>s_mseq_valid(int seq) const</code>	267

12.44.3.7	<code>install_sequence(sequence *seq, int seqnum)</code>	267
12.44.3.8	<code>inner_start(bool state)</code>	268
12.44.3.8	<code>inner_stop()</code>	268
12.44.3.8	<code>clamp_track(int track) const</code>	268
12.44.3.8	<code>set_key_event(unsigned int keycode, long sequence_slot)</code>	268
12.44.3.8	<code>set_key_group(unsigned int keycode, long group_slot)</code>	269
12.44.4	Friends And Related Function Documentation	269
12.44.4.1	<code>jack_sync_callback</code>	269
12.44.5	Field Documentation	269
12.44.5.1	<code>sm_mc_dummy</code>	269
12.44.5.2	<code>m_playscreen_offset</code>	269
12.44.5.3	<code>m_seqs</code>	269
12.44.5.4	<code>m_playback_mode</code>	270
12.44.5.5	<code>m_beats_per_bar</code>	270
12.44.5.6	<code>m_beat_width</code>	270
12.44.5.7	<code>m_one_measure</code>	270
12.44.5.8	<code>m_left_tick</code>	270
12.44.5.9	<code>m_right_tick</code>	270
12.44.5.10	<code>m_starting_tick</code>	270
12.44.5.11	<code>m_tick</code>	270
12.44.5.12	<code>m_seqs_in_set</code>	270
12.44.5.13	<code>m_max_sets</code>	271
12.44.5.14	<code>m_sequence_count</code>	271
12.44.5.15	<code>m_sequence_max</code>	271
12.44.5.16	<code>m_is_modified</code>	271
12.45	<code>seq64::performcallback</code> Struct Reference	271
12.45.1	Detailed Description	272
12.46	<code>seq64::perfull</code> Class Reference	272
12.46.1	Constructor & Destructor Documentation	275
12.46.1.1	<code>~perfull()</code>	275

12.46.2 Member Function Documentation	276
12.46.2.1 set_guides(int snap, int measure, int beat)	276
12.46.2.2 update_sizes()	276
12.46.2.3 init_before_show()	276
12.46.2.4 fill_background_pixmap()	276
12.46.2.5 draw_progress()	277
12.46.2.6 set_ppqn(int ppqn)	277
12.46.2.7 convert_xy(int x, int y, midipulse &ticks, int &seq)	277
12.46.2.8 convert_x(int x, midipulse &ticks)	277
12.46.2.9 snap_x(int &x)	277
12.46.2.10 draw_sequence_on(int seqnum)	278
12.46.2.11 draw_drawable_row(long y)	278
12.46.2.12 enqueue_draw()	278
12.46.2.13 set_zoom(int z)	278
12.46.2.14 on_realize()	278
12.46.2.15 on_expose_event(GdkEventExpose *ev)	278
12.46.2.16 on_button_press_event(GdkEventButton *ev)	278
12.46.2.17 on_button_release_event(GdkEventButton *ev)	278
12.46.2.18 on_key_press_event(GdkEventKey *ev)	279
12.46.3 Friends And Related Function Documentation	279
12.46.3.1 FruityPerfInput	279
12.46.4 Field Documentation	279
12.46.4.1 m_parent	279
12.46.4.2 m_h_page_increment	279
12.46.4.3 m_v_page_increment	279
12.46.4.4 m_fruity_interaction	279
12.47 seq64::perftime Class Reference	280
12.47.1 Constructor & Destructor Documentation	282
12.47.1.1 perftime(perform &perf, perfedit &parent, Gtk::Adjustment &hadjust, int ppqn=SEQ64_USE_DEFAULT_PPQN)	282
12.47.2 Member Function Documentation	282

12.47.2.1 set_guides(int snap, int measure)	282
12.47.2.2 enqueue_draw()	283
12.47.2.3 on_realize()	283
12.47.2.4 on_expose_event(GdkEventExpose *ev)	283
12.47.2.5 key_press_event(GdkEventKey *ev)	283
12.47.3 Field Documentation	283
12.47.3.1 m_parent	283
12.47.3.2 m_measure_length	283
12.47.3.3 m_left_marker_tick	284
12.47.3.4 m_right_marker_tick	284
12.48seq64::rc_settings Class Reference	284
12.48.1 Member Function Documentation	286
12.48.1.1 home_config_directory() const	286
12.49seq64::rect Class Reference	286
12.50seq64::gui_drawingarea_gtk2::rect Struct Reference	286
12.51seq64::Seq24PerfInput Class Reference	287
12.51.1 Member Function Documentation	288
12.51.1.1 on_button_press_event(GdkEventButton *a_ev, perfroll &roll)	288
12.51.1.2 on_button_release_event(GdkEventButton *a_ev, perfroll &roll)	288
12.51.1.3 on_motion_notify_event(GdkEventMotion *a_ev, perfroll &roll)	288
12.51.1.4 set_adding(bool a_adding, perfroll &roll)	288
12.51.1.5 handle_motion_key(bool is_left, perfroll &roll)	288
12.52seq64::Seq24SeqEventInput Struct Reference	289
12.52.1 Member Function Documentation	289
12.52.1.1 set_adding(bool a_adding, seqevent &ths)	289
12.52.1.2 on_button_press_event(GdkEventButton *a_ev, seqevent &ths)	289
12.52.1.3 on_button_release_event(GdkEventButton *a_ev, seqevent &ths)	290
12.52.1.4 on_motion_notify_event(GdkEventMotion *a_ev, seqevent &ths)	290
12.53seq64::Seq24SeqRollInput Class Reference	290
12.53.1 Member Function Documentation	290

12.53.1.1 set_adding(bool a_adding, seqroll &ths)	290
12.53.1.2 on_button_press_event(GdkEventButton *a_ev, seqroll &ths)	290
12.53.1.3 on_button_release_event(GdkEventButton *a_ev, seqroll &ths)	291
12.54seq64::seqdata Class Reference	291
12.54.1 Constructor & Destructor Documentation	294
12.54.1.1 seqdata(sequence &seq, perform &p, int zoom, Gtk::Adjustment &hadjust)	294
12.54.2 Member Function Documentation	294
12.54.2.1 reset()	294
12.54.2.2 set_zoom(int a_zoom)	294
12.54.2.3 set_data_type(midibyte status, midibyte control)	294
12.54.2.4 idle_redraw()	294
12.54.2.5 update_sizes()	295
12.54.2.6 xy_to_rect(int x1, int y1, int x2, int y2, int &rx, int &ry, int &rw, int &rh)	295
12.54.2.7 on_realize()	295
12.54.2.8 on_expose_event(GdkEventExpose *ev)	295
12.54.2.9 on_button_press_event(GdkEventButton *ev)	295
12.54.2.10on_button_release_event(GdkEventButton *ev)	295
12.54.2.11on_motion_notify_event(GdkEventMotion *ev)	295
12.54.2.12on_scroll_event(GdkEventScroll *ev)	296
12.54.3 Field Documentation	296
12.54.3.1 m_number_w	296
12.54.3.2 m_number_h	296
12.54.3.3 m_number_offset_y	296
12.55seq64::seqedit Class Reference	296
12.55.1 Detailed Description	300
12.55.2 Constructor & Destructor Documentation	301
12.55.2.1 seqedit(perform &perf, sequence &seq, int pos, int ppqn=SEQ64_USE_DEFA↵ ULT_PPQN)	301
12.55.3 Member Function Documentation	301
12.55.3.1 set_zoom(int zoom)	301
12.55.3.2 set_snap(int snap)	301

12.55.3.3 set_note_length(int note_length)	301
12.55.3.4 horizontal_adjust(double step)	302
12.55.3.5 set_measures(int lim)	302
12.55.3.6 apply_length(int bpm, int bw, int measures)	302
12.55.3.7 get_measures()	302
12.55.3.8 set_midi_channel(int midichannel)	302
12.55.3.9 set_midi_bus(int midibus)	302
12.55.3.10 set_scale(int scale)	302
12.55.3.11 set_key(int note)	303
12.55.3.12 set_background_sequence(int seq)	303
12.55.3.13 name_change_callback()	303
12.55.3.14 set_data_type(midibyte status, midibyte control=0)	303
12.55.3.15 fill_top_bar()	303
12.55.3.16 create_menus()	303
12.55.3.17 popup_event_menu()	304
12.55.3.18 popup_midibus_menu()	304
12.55.3.19 popup_sequence_menu()	304
12.55.3.20 popup_tool_menu()	304
12.55.3.21 timeout()	304
12.55.3.22 do_action(int action, int var)	305
12.55.3.23 on_delete_event(GdkEventAny *event)	305
12.55.3.24 on_scroll_event(GdkEventScroll *ev)	305
12.55.3.25 on_key_press_event(GdkEventKey *ev)	305
12.55.4 Field Documentation	305
12.55.4.1 m_initial_snap	305
12.55.4.2 m_zoom	306
12.55.4.3 m_pos	306
12.55.4.4 m_seqkeys_wid	306
12.55.4.5 m_seqtime_wid	306
12.55.4.6 m_seqdata_wid	306

12.55.4.7 m_table	306
12.56seq64::seqevent Class Reference	306
12.56.1 Member Function Documentation	309
12.56.1.1 redraw()	309
12.56.1.2 set_snap(int a_snap)	309
12.56.1.3 set_data_type(midibyte a_status, midibyte a_control)	309
12.56.1.4 update_sizes()	309
12.56.1.5 draw_background()	310
12.56.1.6 draw_pixmap_on_window()	310
12.56.1.7 idle_redraw()	310
12.56.1.8 x_to_w(int a_x1, int a_x2, int &a_x, int &a_w)	310
12.56.1.9 drop_event(midipulse a_tick)	310
12.56.1.10draw_events_on(Glib::RefPtr< Gdk::Drawable > a_draw)	310
12.56.1.11start_paste()	310
12.56.1.12change_horz()	310
12.56.1.13convert_x(int x, midipulse &tick)	310
12.56.1.14convert_t(midipulse ticks, int &x)	311
12.56.1.15snap_x(int &a_x)	311
12.56.1.16on_realize()	311
12.56.1.17on_button_press_event(GdkEventButton *a_ev)	311
12.56.1.18on_button_release_event(GdkEventButton *a_ev)	311
12.56.1.19on_motion_notify_event(GdkEventMotion *a_ev)	311
12.56.1.20on_key_press_event(GdkEventKey *a_p0)	312
12.57seq64::seqkeys Class Reference	312
12.57.1 Member Function Documentation	314
12.57.1.1 set_hint_state(bool a_state)	314
12.57.1.2 draw_key(int a_key, bool a_state)	315
12.57.1.3 on_realize()	315
12.57.1.4 on_button_press_event(GdkEventButton *a_ev)	315
12.57.1.5 on_button_release_event(GdkEventButton *a_ev)	315

12.57.1.6 on_motion_notify_event(GdkEventMotion *a_p0)	315
12.57.1.7 on_enter_notify_event(GdkEventCrossing *p0)	316
12.57.1.8 on_leave_notify_event(GdkEventCrossing *p0)	316
12.57.1.9 on_scroll_event(GdkEventScroll *a_ev)	316
12.57.1.10 on_size_allocate(Gtk::Allocation &)	316
12.58 seq64::seqmenu Class Reference	316
12.58.1 Detailed Description	319
12.58.2 Constructor & Destructor Documentation	319
12.58.2.1 seqmenu(perform &a_p)	319
12.58.2.2 ~seqmenu()	319
12.58.3 Member Function Documentation	320
12.58.3.1 is_edit_sequence(int seqnum) const	320
12.58.3.2 popup_menu()	320
12.58.3.3 seq_edit()	320
12.58.3.4 seq_event_edit()	320
12.58.3.5 seq_new()	320
12.58.3.6 seq_copy()	320
12.58.3.7 seq_cut()	321
12.58.3.8 seq_paste()	321
12.58.3.9 seq_clear_perf()	321
12.58.4 Field Documentation	321
12.58.4.1 m_seqedit	321
12.58.4.2 m_modified	321
12.59 seq64::seqroll Class Reference	321
12.59.1 Constructor & Destructor Documentation	325
12.59.1.1 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)	325
12.59.1.2 ~seqroll()	326
12.59.2 Member Function Documentation	326
12.59.2.1 set_zoom(int zoom)	326

12.59.2.2 set_key(int key)	326
12.59.2.3 set_scale(int scale)	326
12.59.2.4 set_data_type(midibyte status, midibyte control)	326
12.59.2.5 set_background_sequence(bool state, int seq)	326
12.59.2.6 update_sizes()	326
12.59.2.7 update_background()	327
12.59.2.8 draw_events_on_pixmap()	327
12.59.2.9 draw_progress_on_window()	327
12.59.2.10 reset()	327
12.59.2.11 redraw()	327
12.59.2.12 follow_progress()	327
12.59.2.13 horizontal_adjust(double step)	327
12.59.2.14 snap_y(int &y)	328
12.59.2.15 snap_x(int &x)	328
12.59.2.16 convert_tn(midipulse ticks, int note, int &x, int &y)	328
12.59.2.17 xy_to_rect(int x1, int y1, int x2, int y2, int &x, int &y, int &w, int &h)	328
12.59.2.18 draw_events_on(Glib::RefPtr<Gdk::Drawable> draw)	328
12.59.2.19 change_horz()	328
12.59.2.20 on_key_press_event(GdkEventKey *ev)	329
12.59.2.21 on_scroll_event(GdkEventScroll *a_ev)	329
12.59.3 Field Documentation	329
12.59.3.1 m_pos	329
12.60 seq64::seqtime Class Reference	329
12.60.1 Constructor & Destructor Documentation	331
12.60.1.1 seqtime(sequence &seq, perform &p, int zoom, Gtk::Adjustment &hadjust, int ppqn=SEQ64_USE_DEFAULT_PPQN)	331
12.60.2 Member Function Documentation	332
12.60.2.1 update_pixmap()	332
12.60.2.2 on_realize()	332
12.60.2.3 on_button_press_event(GdkEventButton *)	332
12.60.2.4 on_button_release_event(GdkEventButton *)	332

12.61 seq64::sequence Class Reference	332
12.61.1 Detailed Description	340
12.61.2 Member Enumeration Documentation	341
12.61.2.1 select_action_e	341
12.61.3 Member Function Documentation	341
12.61.3.1 partial_assign(const sequence &rhs)	341
12.61.3.2 event_count() const	341
12.61.3.3 push_undo()	341
12.61.3.4 pop_undo()	341
12.61.3.5 pop_redo()	341
12.61.3.6 push_trigger_undo()	341
12.61.3.7 set_beats_per_bar(int beatspermeasure)	341
12.61.3.8 set_beat_width(int beatwidth)	342
12.61.3.9 get_beat_width() const	342
12.61.3.10 set_rec_vol(int rec_vol)	342
12.61.3.11 get_name() const	342
12.61.3.12 set_length(midipulse len, bool adjust_triggers=true)	342
12.61.3.13 get_last_tick()	343
12.61.3.14 set_last_tick(midipulse tick)	343
12.61.3.15 mod_last_tick()	343
12.61.3.16 set_playing(bool)	343
12.61.3.17 toggle_queued()	343
12.61.3.18 off_queued()	343
12.61.3.19 set_recording(bool)	343
12.61.3.20 set_snap_tick(int st)	343
12.61.3.21 set_quantized_rec(bool qr)	344
12.61.3.22 set_thru(bool)	344
12.61.3.23 s_dirty_main()	344
12.61.3.24 s_dirty_edit()	344
12.61.3.25 s_dirty_perf()	344

12.61.3.26	<code>s_dirty_names()</code>	344
12.61.3.27	<code>set_dirty_mp()</code>	344
12.61.3.28	<code>set_dirty()</code>	344
12.61.3.29	<code>set_midi_channel(midibyte ch)</code>	344
12.61.3.30	<code>print()</code> const	344
12.61.3.31	<code>print_triggers()</code> const	345
12.61.3.32	<code>play(midipulse tick, bool playback_mode)</code>	345
12.61.3.33	<code>add_event(const event &er)</code>	345
12.61.3.34	<code>add_trigger(midipulse tick, midipulse len, midipulse offset=0, bool adjust_↔ offset=true)</code>	346
12.61.3.35	<code>split_trigger(midipulse tick)</code>	346
12.61.3.36	<code>grow_trigger(midipulse tick_from, midipulse tick_to, midipulse len)</code>	346
12.61.3.37	<code>del_trigger(midipulse tick)</code>	346
12.61.3.38	<code>get_trigger_state(midipulse tick)</code>	346
12.61.3.39	<code>select_trigger(midipulse tick)</code>	346
12.61.3.40	<code>unselect_triggers()</code>	347
12.61.3.41	<code>intersect_triggers(midipulse position, midipulse &start, midipulse &ender)</code>	347
12.61.3.42	<code>intersect_notes(midipulse position, midipulse position_note, midipulse &start, midipulse &ender, int &note)</code>	347
12.61.3.43	<code>intersect_events(midipulse posstart, midipulse posend, midibyte status, midipulse &start)</code>	348
12.61.3.44	<code>paste_trigger()</code>	348
12.61.3.45	<code>move_selected_triggers_to(midipulse tick, bool adjust_offset, int which=2)</code>	348
12.61.3.46	<code>selected_trigger_start()</code>	349
12.61.3.47	<code>selected_trigger_end()</code>	349
12.61.3.48	<code>get_max_trigger()</code>	349
12.61.3.49	<code>move_triggers(midipulse start_tick, midipulse distance, bool direction)</code>	349
12.61.3.50	<code>copy_triggers(midipulse start_tick, midipulse distance)</code>	349
12.61.3.51	<code>clear_triggers()</code>	349
12.61.3.52	<code>set_midi_bus(char mb)</code>	349
12.61.3.53	<code>set_master_midi_bus(mastermidibus *mmb)</code>	349

12.61.3.54	<code>select_note_events(midipulse tick_s, int note_h, midipulse tick_f, int note_l, select_action_e action)</code>	350
12.61.3.55	<code>select_events(midipulse tick_s, midipulse tick_f, midibyte status, midibyte cc, select_action_e action)</code>	350
12.61.3.56	<code>select_events(midibyte status, midibyte cc, bool inverse=false)</code>	350
12.61.3.57	<code>get_num_selected_notes() const</code>	350
12.61.3.58	<code>get_num_selected_events(midibyte status, midibyte cc) const</code>	350
12.61.3.59	<code>select_all()</code>	350
12.61.3.60	<code>copy_selected()</code>	351
12.61.3.61	<code>cut_selected(bool copyevents=true)</code>	351
12.61.3.62	<code>paste_selected(midipulse tick, int note)</code>	351
12.61.3.63	<code>get_selected_box(midipulse &tick_s, int &note_h, midipulse &tick_f, int &note_l)</code>	351
12.61.3.64	<code>get_clipboard_box(midipulse &tick_s, int &note_h, midipulse &tick_f, int &note_l)</code>	351
12.61.3.65	<code>move_selected_notes(midipulse deltatick, int deltanote)</code>	351
12.61.3.66	<code>add_note(midipulse tick, midipulse len, int note, bool paint=false)</code>	351
12.61.3.67	<code>add_event(midipulse tick, midibyte status, midibyte d0, midibyte d1, bool paint=false)</code>	352
12.61.3.68	<code>stream_event(event &ev)</code>	352
12.61.3.69	<code>change_event_data_range(midipulse tick_s, midipulse tick_f, midibyte status, midibyte cc, int d_s, int d_f)</code>	352
12.61.3.70	<code>increment_selected(midibyte status, midibyte control)</code>	353
12.61.3.71	<code>decrement_selected(midibyte status, midibyte control)</code>	353
12.61.3.72	<code>grow_selected(midipulse deltatick)</code>	353
12.61.3.73	<code>stretch_selected(midipulse deltatick)</code>	354
12.61.3.74	<code>remove_marked()</code>	354
12.61.3.75	<code>mark_selected()</code>	354
12.61.3.76	<code>unpaint_all()</code>	354
12.61.3.77	<code>unselect()</code>	354
12.61.3.78	<code>verify_and_link()</code>	354
12.61.3.79	<code>link_new()</code>	354
12.61.3.80	<code>zero_markers()</code>	354
12.61.3.81	<code>play_note_on(int note)</code>	355

12.61.3.82	<code>play_note_off(int note)</code>	355
12.61.3.83	<code>off_playing_notes()</code>	355
12.61.3.84	<code>pause()</code>	355
12.61.3.85	<code>reset(bool live_mode)</code>	355
12.61.3.86	<code>reset_draw_marker()</code>	355
12.61.3.87	<code>reset_draw_trigger_marker()</code>	355
12.61.3.88	<code>get_next_note_event(midipulse *tick_s, midipulse *tick_f, int *note, bool *selected, int *velocity)</code>	355
12.61.3.89	<code>get_minmax_note_events(int &lowest, int &highest)</code>	356
12.61.3.90	<code>get_next_event(midibyte status, midibyte cc, midipulse *tick, midibyte *d0, midibyte *d1, bool *selected)</code>	356
12.61.3.91	<code>get_next_event(midibyte *status, midibyte *cc)</code>	356
12.61.3.92	<code>fill_container(midi_container &c, int tracknumber)</code>	356
12.61.3.93	<code>transpose_notes(int steps, int scale)</code>	356
12.61.3.94	<code>background_sequence(int bs)</code>	357
12.61.3.95	<code>copy_events(const event_list &newevents)</code>	357
12.61.3.96	<code>set_parent(perform *p)</code>	357
12.61.3.97	<code>put_event_on_bus(event &ev)</code>	357
12.61.3.98	<code>set_trigger_offset(midipulse trigger_offset)</code>	357
12.61.3.99	<code>split_trigger(trigger &trig, midipulse splittick)</code>	358
12.61.3.100	<code>adjust_trigger_offsets_to_length(midipulse newlen)</code>	358
12.61.3.101	<code>remove(event_list::iterator i)</code>	358
12.61.3.102	<code>move(event &e)</code>	358
12.61.3.103	<code>move_all()</code>	358
12.61.4	Field Documentation	358
12.61.4.1	<code>m_parent</code>	358
12.61.4.2	<code>m_midi_channel</code>	359
12.61.4.3	<code>m_playing_notes</code>	359
12.61.4.4	<code>m_raise</code>	359
12.61.4.5	<code>m_seq_number</code>	359
12.61.4.6	<code>m_length</code>	359

12.61.4.7 m_snap_tick	359
12.61.4.8 m_time_beats_per_measure	359
12.61.4.9 m_time_beat_width	359
12.61.4.10m_clocks_per_metronome	359
12.61.4.11m_32nds_per_quarter	359
12.61.4.12m_us_per_quarter_note	360
12.61.4.13m_musical_key	360
12.61.4.14m_musical_scale	360
12.61.4.15m_background_sequence	360
12.61.4.16m_mutex	360
12.62seq64::trigger Class Reference	360
12.62.1 Detailed Description	361
12.62.2 Member Function Documentation	361
12.62.2.1 operator<(const trigger &rhs)	361
12.63seq64::triggers Class Reference	361
12.63.1 Constructor & Destructor Documentation	364
12.63.1.1 triggers(sequence &parent)	364
12.63.2 Member Function Documentation	364
12.63.2.1 operator=(const triggers &rhs)	364
12.63.2.2 set_length(int len)	364
12.63.2.3 print(const std::string &seqname) const	364
12.63.2.4 play(midipulse &starttick, midipulse &endtick)	364
12.63.2.5 add(midipulse tick, midipulse len, midipulse offset=0, bool adjustoffset=true)	365
12.63.2.6 adjust_offsets_to_length(midipulse newlen)	365
12.63.2.7 split(midipulse tick)	365
12.63.2.8 split(trigger &trig, midipulse splittick)	366
12.63.2.9 grow(midipulse tickfrom, midipulse tickto, midipulse length)	366
12.63.2.10remove(midipulse tick)	366
12.63.2.11get_state(midipulse tick)	366
12.63.2.12select(midipulse tick)	367

12.63.2.13	<code>unselect()</code>	367
12.63.2.14	<code>intersect(midipulse position, midipulse &start, midipulse &end)</code>	367
12.63.2.15	<code>paste()</code>	367
12.63.2.16	<code>move_selected(midipulse tick, bool adjustoffset, int which=2)</code>	368
12.63.2.17	<code>get_selected_start()</code>	368
12.63.2.18	<code>get_selected_end()</code>	368
12.63.2.19	<code>get_maximum()</code>	368
12.63.2.20	<code>move(midipulse starttick, midipulse distance, bool direction)</code>	368
12.63.2.21	<code>copy(midipulse starttick, midipulse distance)</code>	369
12.63.2.22	<code>next(midipulse *tick_on, midipulse *tick_off, bool *selected, midipulse *tick_offset)</code>	369
12.63.2.23	<code>next_trigger()</code>	370
12.63.2.24	<code>adjust_offset(midipulse offset)</code>	370
12.63.3	Field Documentation	370
12.63.3.1	<code>m_ppqn</code>	370
12.63.3.2	<code>m_length</code>	370
12.64	<code>seq64::user_instrument</code> Class Reference	370
12.64.1	Detailed Description	371
12.64.2	Member Function Documentation	372
12.64.2.1	<code>set_defaults()</code>	372
12.64.2.2	<code>controller_max() const</code>	372
12.64.2.3	<code>controller_name(int c) const</code>	372
12.64.2.4	<code>controller_active(int c) const</code>	372
12.64.2.5	<code>set_controller(int c, const std::string &cname, bool isactive)</code>	372
12.64.2.6	<code>set_name(const std::string &instname)</code>	372
12.64.2.7	<code>copy_definitions(const user_instrument &rhs)</code>	373
12.64.3	Field Documentation	373
12.64.3.1	<code>m_is_valid</code>	373
12.64.3.2	<code>m_controller_count</code>	373
12.65	<code>seq64::user_instrument_t</code> Struct Reference	373
12.65.1	Field Documentation	373

12.65.1.1 instrument	373
12.65.1.2 controllers	373
12.65.1.3 controllers_active	373
12.66seq64::user_midi_bus Class Reference	374
12.66.1 Detailed Description	374
12.66.2 Member Function Documentation	375
12.66.2.1 set_defaults()	375
12.66.2.2 channel_count() const	375
12.66.2.3 channel_max() const	375
12.66.2.4 instrument(int channel) const	375
12.66.2.5 set_instrument(int channel, int instrum)	375
12.66.2.6 copy_definitions(const user_midi_bus &rhs)	375
12.66.3 Field Documentation	376
12.66.3.1 m_is_valid	376
12.66.3.2 m_channel_count	376
12.67seq64::user_midi_bus_t Struct Reference	376
12.67.1 Field Documentation	376
12.67.1.1 alias	376
12.67.1.2 instrument	376
12.68seq64::user_settings Class Reference	376
12.68.1 Detailed Description	383
12.68.2 Member Typedef Documentation	383
12.68.2.1 Busses	383
12.68.2.2 Instruments	383
12.68.3 Member Enumeration Documentation	383
12.68.3.1 mainwid_grid_style_t	383
12.68.4 Member Function Documentation	384
12.68.4.1 set_defaults()	384
12.68.4.2 set_globals() const	384
12.68.4.3 get_globals()	384

12.68.4.4 bus(int index)	384
12.68.4.5 instrument(int index)	384
12.68.4.6 controller_active(int buss, int channel, int cc)	384
12.68.4.7 controller_name(int buss, int channel, int cc)	384
12.68.4.8 zoom(int value)	384
12.68.4.9 mainwnd_rows(int value)	384
12.68.4.10mainwnd_cols(int value)	385
12.68.4.11max_sets(int value)	385
12.68.4.12ext_x(int value)	385
12.68.4.13ext_y(int value)	385
12.68.4.14mainwid_border(int value)	385
12.68.4.15mainwid_spacing(int value)	385
12.68.4.16control_height(int value)	385
12.68.4.17dump_summary()	385
12.68.4.18perf_h_page_increment(int inc)	385
12.68.4.19perf_v_page_increment(int inc)	386
12.68.4.20midi_ppqn(int ppqn)	386
12.68.4.21midi_buss_override(char buss)	386
12.68.4.22midi_beats_per_bar(int beatsperbar)	386
12.68.4.23midi_beats_per_minute(int beatsperminute)	386
12.68.4.24midi_beat_width(int beatwidth)	386
12.68.4.25private_bus(int buss)	386
12.68.4.26private_instrument(int instrum)	386
12.68.5 Field Documentation	386
12.68.5.1 m_midi_buses	386
12.68.5.2 m_instruments	387
12.68.5.3 m_grid_style	387
12.68.5.4 m_grid_brackets	387
12.68.5.5 m_mainwnd_rows	387
12.68.5.6 m_mainwnd_cols	387

12.68.5.7 m_max_sets	387
12.68.5.8 m_mainwid_border	388
12.68.5.9 m_control_height	388
12.68.5.10m_global_seq_feature_save	388
12.68.5.11m_seqedit_scale	388
12.68.5.12m_seqedit_key	388
12.68.5.13m_seqedit_bgsequence	388
12.68.5.14m_use_new_font	388
12.68.5.15m_allow_two_perfedits	389
12.68.5.16m_h_perf_page_increment	389
12.68.5.17m_v_perf_page_increment	389
12.68.5.18m_progress_bar_colored	389
12.68.5.19m_progress_bar_thick	389
12.68.5.20m_window_redraw_rate_ms	389
12.68.5.21m_text_x	389
12.68.5.22m_seqchars_x	390
12.68.5.23m_midi_ppqn	390
12.68.5.24m_midi_beats_per_measure	390
12.68.5.25m_midi_beats_per_minute	390
12.68.5.26m_midi_beat_width	390
12.68.5.27m_midi_buss_override	390
12.68.5.28m_seqs_in_set	390
12.68.5.29m_gmute_tracks	391
12.68.5.30m_max_sequence	391
12.68.5.31m_seqarea_x	391
12.68.5.32m_seqarea_seq_x	391
12.68.5.33m_mainwid_x	391
12.68.5.34m_save_user_config	391
12.68.5.35mc_min_zoom	391
12.68.5.36mc_max_zoom	392
12.68.5.37mc_baseline_ppqn	392
12.69seq64::userfile Class Reference	392
12.69.1 Member Function Documentation	393
12.69.1.1 parse(perform &a_perf)	393
12.69.1.2 write(const perform &a_perf)	393
12.69.1.3 dump_setting_summary()	393

Chapter 1

Sequencer64

Author(s) Chris Ahlstrom 2015-11-27

1.1 Introduction

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

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

Also read the ROADMAP, README, and contrib/bugs_to_investigate files to understand the genesis of this project and the things that still need to be done with Sequencer64.

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/seq24-doc.git>
- <https://github.com/ahlstromcj/sequencer64-doc.git>

These project also have prebuilt PDFs should one not want to bother building them.

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`, standard stuff like the mutex module, the fruity variants (at least the ones already refactored into their own modules), etc., are all left out. Still, the resulting PDF is over 300 pages long.

Some useful references:

- <http://acad.carleton.edu/courses/musc108-00-f14/pages/04/04StandardMIDIFiles.html>
- <http://www.midimusicadventures.com/qs/midi-zips/soundtracks/kq6gm.zip>

Chapter 2

MIDI File Parsing in Sequencer64

Author(s) Chris Ahlstrom 2016-02-13

2.1 Introduction

This section describes the parsing of a MIDI file (and a few other topics). We wanted to add the reading of SMF 0 files to *Sequencer64*. We started with the main format that is supported, SMF 1. Once we understood that we, we figured out how to split a SMF 0 tracks correctly.

We 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 creates one sequence object per channel present in the SMF 0 file, plus the original track. The last pattern slot (sequence 16) 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 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.

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 it's 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.

To summarize the process, here are the relevant event and sequence setter calls typically made while parsing a MIDI track:

1. `perform::add_sequence()`
 - (a) `sequence::sequence()`
 - (b) `sequence::set_master_midi_bus()`
 - (c) `sequence::add_event()`
 - i. `event::event()`
 - ii. `event::set_status()`
 - iii. `event::set_timestamp()`
 - iv. `event::set_data()`
 - (d) `sequence::set_midi_channel()`
 - (e) `sequence::set_length()`
 - (f) `sequence::zero_markers()`
 - (g) `sequence::set_name()`
 - (h) `sequence::set_midi_bus()`
2. `xxxxx::yyyy()`

2.2.3 Meta Events Summary

Here, we summarize the MIDI meta events for your edification.

1. FF 00 02 ssss: Sequence Number.
2. FF 01 len text: Text Event.
3. FF 02 len text: Copyright Notice.
4. FF 03 len text: Sequence/Track Name.
5. FF 04 len text: Instrument Name.
6. FF 05 len text: Lyric.
7. FF 06 len text: Marker.
8. FF 07 len text: Cue Point.
9. FF 08 len text: Patch/program Name.
10. FF 09 len text: Device Name.
11. FF 0A through 0F len text: Other kinds of text events.
12. FF 20 01 cc: MIDI channel (obsolete, used by Cakewalk)
13. FF 21 01 pp: MIDI port (obsolete, used by Cakewalk)
14. FF 2F 00: End of Track.
15. FF 51 03 tttttt: Set Tempo, us/qn.
16. FF 54 05 hr mn se fr ff: SMPTE Offset.
17. FF 58 04 nn dd cc bb: Time Signature.
18. FF 59 02 sf mi: Key Signature.
19. FF 7F len data: Sequencer-Specific.

The next sections describe the events that *Sequencer* tries to handle. These are

- Sequence Number (0x00)
- Track Name (0x03)
- End-of-Track (0x2F)
- Set Tempo (0x51) (Sequencer64 only)
- Time Signature (0x58) (Sequencer64 only)
- Sequencer-Specific (0x7F)
- System Exclusive (0xF0) Sort of handled, functionality incomplete..

2.2.3.1 Sequence Number (0x00)

```
FF 00 02 ss ss
```

This optional event must occur at the beginning of a track, before any non-zero delta-times, and before any transmittable MIDI events. It specifies the number of a sequence.

2.2.3.2 Track/Sequence Name (0x03)

```
FF 03 len text
```

If in a format 0 track, or the first track in a format 1 file, the name of the sequence. Otherwise, the name of the track.

2.2.3.3 End of Track (0x2F)

```
FF 2F 00
```

This event is not optional. It is included so that an exact ending point may be specified for the track, so that it has an exact length, which is necessary for tracks which are looped or concatenated.

2.2.3.4 Set Tempo Event (0x51)

The MIDI Set Tempo meta event sets the tempo of a MIDI sequence in terms of the microseconds per quarter note. This is a meta message, so this event is never sent over MIDI ports to a MIDI device.

After the delta time, this event consists of six bytes of data:

```
FF 51 03 tt tt tt
```

Example:

```
FF 51 03 07 A1 20
```

1. 0xFF is the status byte that indicates this is a Meta event.
2. 0x51 the meta event type that signifies this is a Set Tempo event.
3. 0x03 is the length of the event, always 3 bytes.
4. The remaining three bytes carry the number of microseconds per quarter note. For example, the three bytes above form the hexadecimal value 0x07A120 (500000 decimal), which means that there are 500,000 microseconds per quarter note.

Since there are 60,000,000 microseconds per minute, the event above translates to: set the tempo to $60,000,000 / 500,000 = 120$ quarter notes per minute (120 beats per minute). This is a 24-bit binary value, so each byte covers the full range of 0x00 to 0xFF.

This event normally appears in the first track. If not, the default tempo is 120 beats per minute. This event is important if the MIDI time division is specified in "pulses per quarter note", which does not itself define the length of the quarter note. The length of the quarter note is then determined by the Set Tempo meta event.

Representing tempos as time per beat instead of beat per time allows absolutely exact DWORD-term synchronization with a time-based sync protocol such as SMPTE time code or MIDI time code. This amount of accuracy provided by this tempo resolution allows a four-minute piece at 120 beats per minute to be accurate within 500 usec at the end of the piece.

We have now added the Tempo meta event (and the Time Signature meta event) to the track, which allows other sequencers to obtain these values from a Sequencer64 MIDI file. Here are the original headers for a normal MIDI file and its legacy (Seq24) conversion, as shown by the midicvt application:

```

hymne.asc                                hymne-ppqn-384.asc
MThd 1 4 96                              MThd 1 4 384
MTrk                                      MTrk
0 Meta SeqName "Vangelis: Hymne"        0 SeqNr 0
0 TimeSig 4/4 24 8                      0 Meta SeqName "Vangelis: Hymne"
0 Tempo 750000                          0 SeqSpec 24 24 00 08      (no triggers)
0 Meta TrkEnd                          0 SeqSpec 24 24 00 01 00  (MIDI buss 0)
TrkEnd                                  0 SeqSpec 24 24 00 06 04 04 (beats, width)
                                         0 SeqSpec 24 24 00 02 00  (MIDI ch. 0)
                                         96 Meta TrkEnd
                                         TrkEnd

```

Here is the header data that result from the new conversion, which is used if the "legacy" option is not in force:

```

MThd 1 4 192
MTrk
0 SeqNr 0
0 Meta SeqName "Vangelis: Hymne"
0 TimeSig 4/4 24 8
0 Tempo 750000
0 SeqSpec 24 24 00 08
0 SeqSpec 24 24 00 01 00
0 SeqSpec 24 24 00 06 04 04
0 SeqSpec 24 24 00 02 00
48 Meta TrkEnd
TrkEnd

```

2.2.3.5 Time Signature Event (0x58)

After the delta time, this event consists of seven bytes of data:

```
FF 58 04 nn dd cc bb
```

The time signature is expressed as four numbers. `nn` and `dd` represent the numerator and denominator of the time signature as it would be notated. The numerator counts the number of beats in a measure (beats per measure or beats per bar). The denominator is a negative power of two: 2 represents a quarter-note, 3 represents an eighth-note, etc. The denominator specifies the unit of the beat (e.g. 4 or 8). In Seq24/Sequencer64, this value is also called the "beat width".

The `cc` parameter expresses the number of MIDI clocks (or "ticks", or "pulses") in a metronome click. The standard MIDI clock ticks 24 times per quarter note, so a value of 6 would mean the metronome clicks every 1/8th note. A `cc` value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver). This MIDI clock is different from the clock (PPQN) that determines the start time and duration of the notes.

The `bb` parameter expresses the number of notated 32nd-notes in a MIDI quarter note (24 MIDI Clocks). The usual value for this parameter is 8, though some sequencers allow the user to specify that what MIDI thinks of as a quarter note, should be notated as something else. For example, a value of 16 means that the music plays two quarter notes for each quarter note metered out by the MIDI clock, so that the music plays at double speed.

Examples:

```
FF 58 04 04 02 18 08
```

1. 0xFF is the status byte that indicates this is a Meta event.
2. 0x58 the meta event type that signifies this is a Time Signature event.

3. 0x04 is the length of the event, always 4 bytes.
4. 0x04 is the numerator of the time signature, and ranges from 0x00 to 0xFF.
5. 0x02 is the log base 2 of the denominator, and is the power to which 2 must be raised to get the denominator. Here, the denominator is 2 to 0x02, or 4, so the time signature is 4/4.
6. 0x18 is the metronome pulse in terms of the number of MIDI clock ticks per click. Assuming 24 MIDI clocks per quarter note, the value here (0x18 = 24) indicates that the metronome will tick every 24/24 quarter note. If the value of the sixth byte were 0x30 = 48, the metronome clicks every two quarter notes, i.e. every half-note.
7. 0x08 defines the number of 32nd notes per beat. This byte is usually 8 as there is usually one quarter note per beat, and one quarter note contains eight 32nd notes.

A time signature of 6/8, with a metronome click every 3rd 1/8 note, would be encoded:

```
FF 58 04 06 03 24 08
```

Remember, a 1/4 note is 24 MIDI Clocks, therefore a bar of 6/8 is 72 MIDI Clocks. Hence 3 1/8 notes is 36 (=0x24) MIDI Clocks.

There should generally be a Time Signature Meta event at the beginning of a track (at time = 0), otherwise a default 4/4 time signature will be assumed. Thereafter they can be used to effect an immediate time signature change at any point within a track.

For a format 1 MIDI file, Time Signature Meta events should only occur within the first MTrk chunk.

If a time signature event is not present in a MIDI sequence, 4/4 signature is assumed.

In *Sequencer64*, the `c_timesig SeqSpec` event is given priority. The conventional time signature is used only if the `c_timesig SeqSpec` is not present in the file. NEEDS TO BE TESTED.

2.2.3.6 SysEx Event (0xF0)

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

```
F0 len data F7
```

Sequencer64 has some code in place to store these messages, but the data is currently not actually stored or used. 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 events have been encountered, and those are detected and skipped as well.

2.2.3.7 Sequencer Specific (0x7F)

This data, also known as SeqSpec data, provides a way to encode information that a specific sequencer application needs, while marking it so that other sequences can safely ignore the information.

FF 7F len data

In *Seq24* and *Sequencer64*, the data portion starts with four bytes that indicate the kind of data for a particular SeqSpec event:

c_midibus	^	0x24240001	Track buss number
c_midich	^	0x24240002	Track channel number
c_midiclocks	*	0x24240003	Track clocking
c_triggers	^	0x24240004	See c_triggers_new
c_notes	*	0x24240005	Song data, notes
c_timesig	^	0x24240006	Track time signature
c_bpmtag	*	0x24240007	Song beats/minute
c_triggers_new	^	0x24240008	Track trigger data
c_mutegroups	*	0x24240009	Song mute group data
c_midictrl	*	0x24240010	Song MIDI control
c_musickey	+	0x24240011	Track key (Sequencer64 only)
c_musicscale	+	0x24240012	Track scale (Sequencer64 only)
c_backsequence	+	0x24240013	Track background sequence (Sequencer64 only)

* = global only; ^ = track only; + = both

In *Seq24*, these events are placed at the end of the song, but are not marked as SeqSpec data. Most MIDI applications handle this situation fine, but some (e.g. midicvt) do not. Therefore, *Sequencer64* makes sure to wrap each data item in the 0xFF 0x7F wrapper.

Also, the last three items above (key, scale, and background sequence) can also be stored (by *Sequencer64*) with a particular sequence/track, as well as at the end of the song. Not sure if this bit of extra flexibility is useful, but it is there.

2.2.3.8 Non-Specific End of Sequence

Any other statuses are deemed unsupportable in *Sequencer64*, 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.

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

This code basically works. For now, please look at the source code for more details. Also, the reading of SMF 0 MIDI files is described in the *sequencer64-doc* project on GitHub.

2.4 Running Status

When we apply the `midicvt` application to a file saved by *Sequencer64*, we can end up with a successful ASCII conversion that ends with an error message:

```
$ midicvt hymne-seq64.midi -o hymne-seq64.asc
? Error at MIDI file offset 12155 [0x2f7b]
Error: Garbage at end 'readtrack(): unexpected running status'
```

Is this a problem in `midicvt` or *Sequencer4*? Let's learn about running status.

Running status is a way to speed up the sending of MIDI bytes to a synthesizer or sequencer by taking advantage of redundancy where possible. For example, if we're sending a consecutive group of Note On and Note Off messages to a particular channel, we can save some time by not sending the channel status byte after the first time. Here's an example with Note On on channel 1:

```
0x90 3C 7F
0x90 40 7F
0x90 43 F3
```

Since no change in status occurs after the first of these three events, we can drop the subsequent status bytes:

```
0x90 3C 7F
40 7F
43 F3
```

The 0x90 byte is saved in a "running status buffer" (RSB), and is filled in by the receiving device.

Here is the sequence of events for operating with running status.

1. Clear the RSB buffer (RSB = 0) to start.
2. If a **Voice Category Status** (VCS) byte is received, then set RSB = VCS. VCS bytes range from 0x80 to 0xEF. This is binary 1000000 to 11100000.
3. If a data byte is received (data bytes range from 0x00 to 0x7F, binary 0000000 to 0111111; that is, bit 7 is always 0 in a data byte):
 - (a) If RSB != 0, first insert the RSB into the incoming data stream, then insert the data byte.
 - (b) If RSB == 0, then just insert the data byte into the incoming data stream.
4. Clear the RSB buffer (RSB = 0) when a System Common Message (SCM) status byte is received. SCM bytes range from 0xF0 to 0xF7.
5. The message after an SCM **must** begin with a status byte. That is a byte with bit 7 set.
6. Do no special action when a Realtime Category Message (RCM) byte is received. RCM bytes range from 0xF8 to 0xFF.

Note that some events, such as Tempo, assume that its bytes are all data bytes.

Chapter 3

JACK, Live, and Song Modes in Sequencer64

Author(s) Chris Ahlstrom 2016-01-23

3.1 Introduction

This section describes the interactions between JACK settings and the Live/Song Mode settings, with an eye to describing the proper behavior of Sequencer64 with JACK settings, how the Live/Song modes are supposed to work, and what bugs or issues remain in Sequencer64's JACK handling.

I'm not sure why Doxygen is applying the "code" font so often here. Weird, annoying.

3.2 JACK Functions

Please study the following URL and note these important points:

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

- The timebase master continuously updates position information, beats, timecode, etc. There is at most one master active at a time. If no client is registered as timebase master, frame numbers will be the only position information available.
- The timebase master registers a callback that updates position information while transport is rolling. Its output affects the following process cycle. This function is called immediately after the process callback in the same thread whenever the transport is rolling, or when any client has set a new position in the previous cycle.
- Clients that don't declare a sync callback are assumed ready immediately, anytime the transport wants to start. If a client doesn't require slow-sync processing, it can set its sync callback to NULL.
- The transport state is always valid; initially it is JackTransportStopped.
- When someone calls `jack_transport_start()`, the engine resets the poll bits and changes to a new state, JackTransportStarting.
- When all slow-sync clients are ready, the state changes to JackTransportRolling.

Does Sequencer64 need a latency callback?

http://jackaudio.org/files/docs/html/group__ClientCallbacks.html

(We need to see why most of the following is in a monospaced font. Is there a new Doxygen feature?)

Here are summaries of the JACK functions used in the `jack_assistant` module:

3.2.1 jack_client_open()

Open a client session with a JACK server. More complex and powerful than `jack_client_new()`. Clients choose which of several servers to connect, and how to start the server automatically, if not already running. There is also an option for JACK to generate a unique client name.

```
const char *    client_name,
jack_options_t  options,
jack_status_t * status,
...
```

`client_name` of at most `jack_client_name_size()` characters. The name scope is local to each server. Unless forbidden by the `JackUseExactName` option, the server will modify this name to create a unique variant, if needed.

`options` formed by OR-ing together `JackOptions` bits. Only the `JackOpenOptions` bits are allowed.

`status` (if non-NULL) an address for JACK to return information from the open operation. This status word is formed by OR-ing together the relevant `JackStatus` bits.

Optional parameters: depending on corresponding [options bits] additional parameters may follow `status` (in this order).

[`JackServerName`] (`char *`) `server_name` selects from among several possible concurrent server instances. Server names are unique to each user. If unspecified, use "default" unless `$JACK_DEFAULT_SERVER` is defined in the process environment.

Returns:

Opaque client handle if successful. If this is NULL, the open operation failed, and `*status` includes `JackFailure`, and the caller is not a JACK client.

3.2.2 jack_on_shutdown()

Registers a function to call when the JACK server shuts down the client thread. It must be an asynchronous POSIX signal handler: only async-safe functions, executed from another thread. A typical function might set a flag or write to a pipe so that the rest of the application knows that the JACK client thread has shut down. Clients do not need to call this function. It only helps clients understand what is going on. It should be called before `jack_client_activate()`.

3.2.3 jack_set_sync_callback()

Register/unregister as a slow-sync client; it can't respond immediately to transport position changes. The callback is run at the first opportunity after registration: if the client is active, this is the next process cycle, otherwise it is the first cycle after `jack_activate()`. After that, it runs as per `JackSyncCallback` rules. Clients that don't set this callback are assumed ready immediately any time the transport wants to start.

3.2.4 jack_set_process_callback()

Tells the JACK server to call the callback whenever there is work. The function must be suitable for real-time execution, it cannot call functions that might block for a long time: `malloc()`, `free()`, `printf()`, `pthread_mutex_lock()`, `sleep()`, `wait()`, `poll()`, `select()`, `pthread_join()`, `pthread_cond_wait()`, etc. In the current class, this function is a do-nothing function.

3.2.5 jack_set_session_callback()

Tells the JACK server to call the callback when a session event is delivered. Setting more than one session callback per process is probably a design error. For a multiclient application, it's more sensible to create a JACK client with only one session callback.

3.2.6 jack_activate()

Tells the JACK server that the application is ready to start processing.

3.2.7 jack_release_timebase()

TODO

3.2.8 jack_client_close()

TODO

3.2.9 jack_transport_start()

Starts the JACK transport rolling. Any client can make this request at any time. It takes effect no sooner than the next process cycle, perhaps later if there are slow-sync clients. This function is realtime-safe. No return code.

3.2.10 jack_transport_stop()

Starts the JACK transport rolling. Any client can make this request at any time. This function is realtime-safe. No return code.

3.2.11 jack_transport_locate()

Repositions the transport to a new frame number. May be called at any time by any client. The new position takes effect in two process cycles. If there are slow-sync clients and the transport is already rolling, it will enter the `JackTransportStarting` state and begin invoking their `sync_callbacks` until ready. This function is realtime-safe.

3.2.12 jack_transport_reposition()

Request a new transport position. May be called at any time by any client. The new position takes effect in two process cycles. If there are slow-sync clients and the transport is already rolling, it will enter the JackTransportStarting state and begin invoking their sync_callbacks until ready. This function is realtime-safe. This call, made in the position() function, is currently disabled.

3.2.13 jack_transport_query()

Query the current transport state and position. This function is realtime-safe, and can be called from any thread. If called from the process thread, pos corresponds to the first frame of the current cycle and the state returned is valid for the entire cycle.

The first parameter is the client, which is a pointer to the JACK client structure.

The second parameter is a pointer to structure for returning current transport position; pos->valid will show which fields contain valid data. If pos is NULL, do not return position information.

This function returns the current transport state.

3.3 Modes Operation

3.3.1 No JACK, Live Mode

In `~/ .config/sequencer64/sequencer64.rc`, set:

- `jack_transport = 0`
- `jack_master = 0`
- `jack_master_cond = 0`
- `jack_start_mode = 0`

By changing the start mode to 0 (false), Sequencer64 is put into Live Mode. With this setting, control of the muting and unmuting of patterns resides in the main window (the patterns window). One can start the playback in the performance (song) window, but it will not affect which patterns play, at all.

Note that this option is part of the *File / Options / JACK/LASH* configuration page.

3.3.2 No JACK, Song Mode

In `~/ .config/sequencer64/sequencer64.rc`, set:

- `jack_transport = 0`
- `jack_master = 0`
- `jack_master_cond = 0`
- `jack_start_mode = 1`

By changing the start mode to 1 (true), Sequencer64 is put into Song Mode.

With this setting, control of the muting and unmuting of patterns resides in the song window (the performance window). The patterns shown in the pattern slots of the main window turn on and off whenever the progress bar is in the pattern as drawn in the performance window.

Note that this option is part of the *File / Options / JACK/LASH* configuration page.

3.3.3 JACK Transport

In `~/ .config/sequencer64/sequencer64.rc`, set:

- `jack_transport = 1`
- `jack_master = 0`
- `jack_master_cond = 0`
- `jack_start_mode = 0` or `1` (see previous section)

The current behavior is that `qjackctl` and `sequencer64` playback/progress seem to be independent of each other.

The workaround seems to be to set `seq24/sequencer64` as JACK Master, or if another *application* (e.g. `Qtractor`) is JACK Master.

OLD BEHAVIOR:

Start `qjackctl`, verify that it sets up correctly, then click it's "play" button to start the transport rolling. Run `sequencer64`, load a file. Then note that starting playback (whether in the main window or in the performance window) is ineffective, but resets the time counter in `qjackctl`. Why? With JACK sync enabled by the macro:

```
[JACK transport slave]
jack sync(): zero frame rate [single report]!?
[JackTransportRolling]
[JackTransportStarting] (every time space bar pressed)
[Start playback]
. . .
```

END OF OLD BEHAVIOR.

3.4 Breakage

Old message about `seq24` being broken:

<http://lists.linuxaudio.org/pipermail/linux-audio-user/2010-November/073848.html>

```
i dont see the transport synchronisation working with a jack1 svn version.
you are still using only a sync callback.
```

```
and you are relying on the transport to go through the
JackTransportStarting state.
```

```
this issue should be fixed.
iirc we came to the conclusion, that seq24 is broken, and we will not
revert the changes in jack, which break it.
```

```
the quick and dirty fix on your side, would be to register an empty
process_callback.
```

```
but the issue still remains. seq24 is NOT a slow sync client. but it
registers a sync_callback.
and it even takes a lock in the sync callback.
```

```
the patch for jack-session support didnt get merged either.
```

Another one (no need for a URL):

```
I use seq24 for the majority of my projects but it isn't ideal (I should
point out that I never finish anything). I don't like seq24's pianoroll
editor, the way you do CC envelopes isn't ideal, it uses alsa-midi, there's
unnecessary complexity in switching from pattern-trigger mode to song mode,
and its insistence on being transport master while not even being able to
adjust tempo when live is annoying
```

3.5 JACK References

- <http://libremusicproduction.com/articles/demystifying-jack-%E2%80%93-beginners-guide>
- <http://jackaudio.org/files/docs/html/transport-design.html>
- <http://kxstudio.linuxaudio.org/Repositories>

Chapter 4

User Testing of Sequencer64 with Yoshimi

Author(s) Chris Ahlstrom 2016-03-04

4.1 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.

Please note that the most advanced and recent testing can be found currently in the document `contrib/notes/jack-testing.txt`. We will eventually merge the final tests here... someday.

4.2 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 sequencer64 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/sequencer64-doc.git>

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

4.3 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.

4.3.1 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.

4.3.2 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.

4.3.3 The Sequencer64 User File

To be discussed.

4.4 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.

4.4.1 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>

4.4.2 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!)

4.4.2.1 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.

4.5 Specific Fault Debugging

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

4.6 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.mid`, 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 5

Licenses

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

Author(s) Chris Ahlstrom 2015-09-10

5.1 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.

5.2 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>

5.3 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>

5.4 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>

5.5 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.

5.6 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 6

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 `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::editable_events::save_events ()`

Consider what to do about the `sequence::m_is_modified` flag.

Global `seq64::eventedit::handle_save ()`

Could also support writing the events to a new sequence, for added flexibility.

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

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

Global `seq64::mainwnd::mainwnd (perform &a_p, bool allowperf2=true, int ppqn=SEQ64_USE_DEFAULT_PPQN)`

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::perfedit::perfedit (perform &p, bool second_perfedit=false, 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) const`

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_left_tick (midipulse tick, bool setstart=true)`

The `perform::m_one_measure` member is currently hardwired to `PPQN * 4`.

Global `seq64::pulses_to_string` (midipulse p)

Still needs to be unit tested.

Global `seq64::pulses_to_timestring` (midipulse p, const `midi_timing` &timinginfo)

Still needs to be unit tested.

Global `seq64::seqdata::on_scroll_event` (GdkEventScroll *ev)

DOCUMENT the seqdata scrolling behavior in the documentation projects.

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::m_modified`

We need to make sure that the perform object is in control of the modification flag.

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::seqroll::follow_progress` ()

If playback is disabled (such as by a trigger), then do not update the page;

- When it comes back, make sure we're on the correct page;
- When it stops, put the window back to the beginning, even if the beginning is not defined as "0".

Global `seq64::sequence::get_minmax_note_events` (int &lowest, int &highest)

For efficiency, we should calculate this only when the event set changes, and save the results and return them if good.

Global `seq64::sequence::remove` (event &e)

Use find instead in `sequence::remove()`!

Global `seq64::triggers::next` (midipulse *tick_on, midipulse *tick_off, bool *selected, midipulse *tick_offset)

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

Chapter 7

Deprecated List

Global [seq64::clock_tick_duration_bogus](#) (int bpm, int ppqn)

This is a somewhat bogus calculation used only for "statistical" output in the old perform module. Name changed to reflect this unfortunate fact. Use [pulse_length_us\(\)](#) instead.

Global [seq64::sequence::get_name](#) () const

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

[seq64](#)

Define this macro to use the new seq24 v ??

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

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

seq64::AbstractPerfInput	71
seq64::Seq24PerfInput	287
seq64::automutex	72
seq64::click	72
seq64::configfile	76
seq64::optionsfile	230
seq64::userfile	392
seq64::editable_events	87
seq64::event	91
seq64::editable_event	78
seq64::event_list::event_key	101
seq64::event_list	103
seq64::font	127
seq64::gui_assistant	131
seq64::gui_assistant_gtk2	132
seq64::gui_palette_gtk2	144
seq64::gui_drawingarea_gtk2	134
seq64::eventslots	115
seq64::maintime	169
seq64::mainwid	174
seq64::perfnames	239
seq64::perfroll	272
seq64::perftime	280
seq64::seqdata	291
seq64::seqevent	306
seq64::seqkeys	312
seq64::seqroll	321
seq64::seqtime	329
seq64::gui_window_gtk2	146
seq64::eventedit	108
seq64::mainwnd	182
seq64::perfedit	233
seq64::seqedit	296

seq64::jack_assistant	148
seq64::jack_scratchpad	157
seq64::keybindentry	157
seq64::keys_perform	159
seq64::keys_perform_gtk2	163
seq64::keys_perform_transfer	165
seq64::keystroke	166
seq64::lash	168
seq64::mastermidibus	191
seq64::midi_container	199
seq64::midi_list	202
seq64::midi_vector	209
seq64::midi_measures	203
seq64::midi_splitter	204
seq64::midi_timing	207
seq64::midibus	210
seq64::midifile	215
seq64::mutex	228
seq64::condition_var	75
seq64::options	229
seq64::perform	243
seq64::performcallback	271
seq64::mainwnd	182
seq64::rc_settings	284
seq64::rect	286
seq64::gui_drawingarea_gtk2::rect	286
seq64::Seq24SeqEventInput	289
seq64::Seq24SeqRollInput	290
seq64::seqmenu	316
seq64::mainwid	174
seq64::perfnames	239
seq64::sequence	332
seq64::trigger	360
seq64::triggers	361
seq64::user_instrument	370
seq64::user_instrument_t	373
seq64::user_midi_bus	374
seq64::user_midi_bus_t	376
seq64::user_settings	376

Chapter 10

Data Structure Index

10.1 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::automutex	Provides a mutex that locks automatically when created, and unlocks when destroyed	??
seq64::click	Encapsulates any possible mouse click	??
seq64::condition_var	A mutex works best in conjunction with a condition variable	??
seq64::configfile	This class is the abstract base class for optionsfile and userfile	??
seq64::editable_event	Provides for the management of MIDI editable events	??
seq64::editable_events	Provides for the management of an ordered collection MIDI editable events	??
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::eventedit	This class supports an Event Editor that is used to tweak the details of events and get a better idea of the mix of events in a sequence	??
seq64::eventslots	This class implements the left-side keyboard in the patterns window	??
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::mastermidibus	The class that "supervises" all of the midibus objects?	??
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_measures	Provides a data structure to hold the numeric equivalent of the measures string "measures↔:beats:divisions" ("m:b:d")	??
seq64::midi_splitter	This class handles the parsing and writing of MIDI files	??
seq64::midi_timing	We anticipate the need to have a small structure holding the parameters needed to calculate MIDI times within an arbitrary song	??
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::mutex	Simple wrapper for the pthread_mutex_t type used as a recursive mutex	??
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::rect	A small helper class representing a rectangle	??
seq64::gui_drawingarea_gtk2::rect	A small helper structure 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 sequences and the configuration of the user-interface ??

[seq64::userfile](#)

Supports the user's `~/ .config/sequencer64/sequencer64.usr` and `~/ .seq24usr` configuration file ??

Chapter 11

Namespace Documentation

11.1 seq64 Namespace Reference

Define this macro to use the new seq24 v.

Data Structures

- class [AbstractPerfInput](#)
Provides an abstract base class to provide the minimal interface for the various "perf input" classes.
- class [automutex](#)
Provides a mutex that locks automatically when created, and unlocks when destroyed.
- class [click](#)
Encapsulates any possible mouse click.
- class [condition_var](#)
A mutex works best in conjunction with a condition variable.
- class [configfile](#)
This class is the abstract base class for optionsfile and userfile.
- class [editable_event](#)
Provides for the management of MIDI editable events.
- class [editable_events](#)
Provides for the management of an ordered collection MIDI editable events.
- class [event](#)
Provides events for management of MIDI events.
- class [event_list](#)
The [event_list](#) class is a receptable for MIDI events.
- class [eventedit](#)
This class supports an Event Editor that is used to tweak the details of events and get a better idea of the mix of events in a sequence.
- class [eventslots](#)
This class implements the left-side keyboard in the patterns window.
- class [font](#)
This class provides a wrapper for rendering fonts that are encoded as a 16 x 16 pixmap file in XPM format.
- class [gui_assistant](#)
This class provides an interface for some of the GUI support needed in Sequencer64.
- class [gui_assistant_gtk2](#)

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

- class [gui_drawingarea_gtk2](#)

Implements the basic drawing areas of the application.

- class [gui_palette_gtk2](#)

Implements a stock palette of Gdk::Color elements.

- class [gui_window_gtk2](#)

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

- class [jack_assistant](#)

This class provides the performance mode JACK support.

- class [jack_scratchpad](#)

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

- class [keybindentry](#)

Class for management of application key-bindings.

- class [keys_perform](#)

This class supports the performance mode.

- class [keys_perform_gtk2](#)

This class supports the performance mode.

- struct [keys_perform_transfer](#)

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

- class [keystroke](#)

Encapsulates any practical keystroke.

- class [lash](#)

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

- class [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.

- class [mainwid](#)

This class implement the piano roll area of the application.

- class [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.

- class [mastermidibus](#)

The class that "supervises" all of the midibus objects?

- class [midi_container](#)

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

- class [midi_list](#)

This class is the std::list implementation of the [midi_container](#).

- class [midi_measures](#)

Provides a data structure to hold the numeric equivalent of the measures string "measures:beats:divisions" ("m:b:d").

- class [midi_splitter](#)

This class handles the parsing and writing of MIDI files.

- class [midi_timing](#)

We anticipate the need to have a small structure holding the parameters needed to calculate MIDI times within an arbitrary song.

- class [midi_vector](#)

This class is the std::vector implementation of the [midi_container](#).

- class [midibus](#)

Provides a class for handling the MIDI buss on Linux.

- class [midifile](#)

This class handles the parsing and writing of MIDI files.

- class [mutex](#)

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

 - class [options](#)

This class supports a full tabbed options dialog.
 - class [optionsfile](#)

Provides a file for reading and writing the application's main configuration file.
 - class [perfedit](#)

This class supports a Performance Editor that is used to arrange the patterns/sequences defined in the patterns panel.
 - class [perfnames](#)

This class implements the left-side keyboard in the patterns window.
 - class [perform](#)

This class supports the performance mode.
 - struct [performcallback](#)

Provides for notification of events.
 - class [perroll](#)

This class implements the performance roll user interface.
 - class [perftime](#)

This class implements drawing the piano time at the top of the "performance window" (the "song editor").
 - class [rc_settings](#)

This class contains the options formerly named "global_XXXXXX".
 - class [rect](#)

A small helper class representing a rectangle.
 - class [Seq24PerfInput](#)

Implements the default (Seq24) performance input characteristics of this application.
 - struct [Seq24SeqEventInput](#)

This structure implement the normal interaction methods for Seq24.
 - class [Seq24SeqRollInput](#)

Implements the Seq24 mouse interaction paradigm for the seqroll.
 - class [seqdata](#)

This class supports drawing piano-roll events on a window.
 - class [seqedit](#)

Implements the Pattern Editor, which has references to:
 - class [segevent](#)

Implements the piano event drawing area.
 - class [seqkeys](#)

This class implements the left side piano of the pattern/sequence editor.
 - class [seqmenu](#)

This class handles the right-click menu of the sequence slots in the pattern window.
 - class [seqroll](#)

Implements the piano roll section of the pattern editor.
 - class [seqtime](#)

This class implements the piano time, whatever that is.
 - class [sequence](#)

The sequence class is firstly a receptable for a single track of MIDI data read from a MIDI file or edited into a pattern.
 - class [trigger](#)

This class hold a single trigger for a sequence object.
 - class [triggers](#)

The triggers class is a receptable the triggers that can be used with a sequence object.
 - class [user_instrument](#)

Provides data about the MIDI instruments, readable from the "user" configuration file.
 - struct [user_instrument_t](#)

This structure corresponds to [user-instrument-N] definitions in the `~/seq24usr` or `~/config/sequencer64/sequsr` file.

- class `user_midi_bus`
Provides data about the MIDI busses, readable from the "user" configuration file.
- struct `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).
- class `user_settings`
Holds the current values of sequence settings and settings that can modify the number of sequences and the configuration of the user-interface.
- class `userfile`
Supports the user's `~/config/sequencer64/sequencer64 usr` and `~/seq24usr` configuration file.

Typedefs

- typedef unsigned char `midibyte`
Provides a fairly common type definition for a byte value.
- typedef unsigned char `bussbyte`
Distinguishes a bus number from other MIDI bytes.
- typedef unsigned short `midishort`
Distinguishes a short value from the unsigned short values implicit in short-valued MIDI numbers.
- typedef unsigned long `midilong`
Distinguishes a long value from the unsigned long values implicit in long-valued MIDI numbers.
- typedef long `midipulse`
Distinguishes a long value from the unsigned long values implicit in MIDI time measurements.

Enumerations

Functions

- bool `extract_timing_numbers` (const std::string &s, std::string &part_1, std::string &part_2, std::string &part_3, std::string &fraction)
Extracts up to 4 numbers from a colon-delimited string.
- std::string `pulses_to_string` (midipulse p)
Converts MIDI pulses (also known as ticks, clocks, or divisions) into a string.
- std::string `pulses_to_measurestring` (midipulse p, const midi_timing &seqparms)
Converts a MIDI pulse/ticks/clock value into a string that represents "measures:beats:ticks" ("measures:beats:division").
- bool `pulses_to_midi_measures` (midipulse p, const midi_timing &seqparms, midi_measures &measures)
Converts a MIDI pulse/ticks/clock value into a string that represents "measures:beats:ticks" ("measures:beats:division").
- std::string `pulses_to_timestring` (midipulse p, int bpm, int ppqn)
Converts a MIDI pulse/ticks/clock value into a string that represents "hours:minutes:seconds.fraction".
- std::string `pulses_to_timestring` (midipulse p, const midi_timing &timinginfo)
Converts a MIDI pulse/ticks/clock value into a string that represents "hours:minutes:seconds.fraction".
- midipulse `measurestring_to_pulses` (const std::string &measures, const midi_timing &seqparms)
Converts a string that represents "measures:beats:division" to a MIDI pulse/ticks/clock value.
- midipulse `midi_measures_to_pulses` (const midi_measures &measures, const midi_timing &seqparms)
Converts a string that represents "measures:beats:division" to a MIDI pulse/ticks/clock value.

- [midipulse timestring_to_pulses](#) (const std::string ×tring, int bpm, int ppqn)
Converts a string that represents "hours:minutes:seconds.fraction" into a MIDI pulse/ticks/clock value.
- [midipulse string_to_pulses](#) (const std::string &s, const [midi_timing](#) &mt)
Converts a time string to pulses.
- [midibyte string_to_midibyte](#) (const std::string &s)
Converts a string to a MIDI byte.
- std::string [shorten_file_spec](#) (const std::string &fpath, int leng)
Shortens a file-specification to make sure it is no longer than the provided length value.
- bool [string_not_void](#) (const std::string &s)
Tests that a string is not empty and has non-space characters.
- bool [string_is_void](#) (const std::string &s)
Tests that a string is empty or has only white-space characters.
- bool [strings_match](#) (const std::string &target, const std::string &x)
Compares two strings for a form of semantic equality, for the purposes of [editable_event\(\)](#), for example.
- int [log2_time_sig_value](#) (int tsd)
Calculates the log-base-2 value of a number that is already a power of 2.
- void [tempo_to_bytes](#) (midibyte t[3], int tempo_us)
Provide a way to convert a tempo value (microseconds per quarter note) into the three bytes needed as value in a Tempo meta event.
- double [beats_per_minute_from_tempo](#) (double tempo)
This function calculates the effective beats-per-minute based on the value of a Tempo meta-event.
- double [tempo_from_beats_per_minute](#) (double bpm)
This function is the inverse of [beats_per_minute_from_tempo\(\)](#).
- double [pulse_length_us](#) (int bpm, int ppqn)
Calculates pulse-length from the BPM (beats-per-minute) and PPQN (pulses-per-quarter-note) values.
- double [delta_time_us_to_ticks](#) (unsigned long us, int bpm, int ppqn)
Converts delta time in microseconds to ticks.
- double [ticks_to_delta_time_us](#) (midipulse delta_ticks, int bpm, int ppqn)
Converts the time in ticks ("clocks") to delta time in microseconds.
- double [clock_tick_duration_bogus](#) (int bpm, int ppqn)
Calculates the duration of a clock tick based on PPQN and BPM settings.
- int [clock_ticks_from_ppqn](#) (int ppqn)
A simple calculation to convert PPQN to MIDI clock ticks.
- double [double_ticks_from_ppqn](#) (int ppqn)
A simple calculation to convert PPQN to MIDI clock ticks.
- [midipulse measures_to_ticks](#) (int bpm, int ppqn, int bw, int measures=1)
Calculates the length of an integral number of measures, in ticks.
- bool [help_check](#) (int argc, char *argv[])
Checks to see if the first option is a help or version argument, just so we can skip the "Reading configuration ..." messages.
- bool [parse_options_files](#) ([perform](#) &p, int argc, char *argv[])
Provides the command-line option support, as well as some setup support, extracted from the main routine of Sequencer64.
- int [parse_command_line_options](#) (int argc, char *argv[])
Parses the command-line options on behalf of the application.
- bool [write_options_files](#) (const [perform](#) &p)
Saves all options to the "rc" and "user" configuration files.
- std::string [to_string](#) (const [event](#) &ev)
A free function to convert an event into an informative string, just enough to save some debugging time.
- bool [file_exists](#) (const std::string &filename)
Checks a file for existence.

- bool [file_readable](#) (const std::string &filename)
Checks a file for readability.
- bool [file_writable](#) (const std::string &filename)
Checks a file for writability.
- bool [file_accessible](#) (const std::string &filename)
Checks a file for readability and writability.
- bool [file_executable](#) (const std::string &filename)
Checks a file for the ability to be executed.
- bool [file_is_directory](#) (const std::string &filename)
Checks a file to see if it is a directory.
- bool [make_directory](#) (const std::string &pathname)
A function to ensure that the ~/.config/sequencer64 directory exists.
- int [choose_ppqn](#) (int ppqn)
Common code for handling PPQN settings.
- bool [ppqn_is_valid](#) (int ppqn)
Common code for handling PPQN settings.
- 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_callback](#) (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)
The JACK timebase function defined here sets the JACK position structure.
- void [jack_session_callback](#) (jack_session_event_t *ev, void *arg)
Set the m_jsession_ev (event) value of the perform object.
- bool [create_lash_driver](#) (perform &p, int argc, char **argv)
Creates and starts a lash object.
- [lash * lash_driver](#) ()
Provides access to the lash object.
- void [delete_lash_driver](#) ()
Deletes the last object.
- void * [output_thread_func](#) (void *p)
Global functions defined in perform.cpp.
- void * [input_thread_func](#) (void *myperf)
Set up the performance, and set the process to realtime privileges.
- long [min](#) (long a, long b)
[min\(\)](#) for long values.
- static std::string [make_section_name](#) (const std::string &label, int value)
Provides a purely internal, ad hoc helper function to create numbered section names for the userfile class.
- [font & font_render](#) ()
The p_font_renderer pointer was once created in the main module, sequencer64.cpp.
- Gtk::Adjustment & [adjustment_dummy](#) ()
Provides a way to provide a dummy Gtk::Adjustment object, but not create one until it is actually needed, so that the Glib/Gtk infrastructure is ready for it.

Variables

- std::string [c_controller_names](#) [SEQ64_MIDI_COUNT_MAX]
Provides the default names of MIDI controllers.
- const midibyte [EVENT_STATUS_BIT](#)
This highest bit of the status byte is always 1.
- const midibyte [EVENT_NOTE_OFF](#)
Channel Voice Messages.
- const midibyte [EVENT_MIDI_SYSEX](#)
System Messages.
- const midibyte [EVENT_MIDI_META](#)
0xFF is a MIDI "escape code" used in MIDI files to introduce a MIDI meta event.
- const midibyte [EVENT_SYSEX](#)
A MIDI System Exclusive (SYSEX) message starts with F0, followed by the manufacturer ID (how many? bytes), a number of data bytes, and ended by an F7.
- const midibyte [EVENT_NULL_CHANNEL](#)
This value of 0xFF is Sequencer64's channel value that indicates that the event's m_channel value is bogus.
- const midibyte [EVENT_GET_CHAN_MASK](#)
These file masks are used to obtain or to mask off the channel data from a status byte.
- const int [c_midibus_output_size](#)
Manifest global constants.
- const midilong [c_midibus](#)
Provides tags used by the midifile class to control the reading and writing of the extra "proprietary" information stored in a Seq24 MIDI file.
- const int [c_midi_track_ctrl](#)
Pseudo control values for associating MIDI events (I think) with automation of some of the controls in seq24.
- const bool [c_scales_policy](#) [c_scale_size][SEQ64_OCTAVE_SIZE]
Each value in the kind of scale is denoted by a true value in these arrays.
- const int [c_scales_transpose_up](#) [c_scale_size][SEQ64_OCTAVE_SIZE]
Increment values needed to transpose each scale up so that it remains in the same key.
- const int [c_scales_transpose_dn_neg](#) [c_scale_size][SEQ64_OCTAVE_SIZE]
Making these positive makes it easier to read.
- const char [c_scales_text](#) [c_scale_size][32]
The names of the supported scales.
- const char [c_key_text](#) [SEQ64_OCTAVE_SIZE][3]
Provides the entries for the Key dropdown menu in the Pattern Editor window.
- const char [c_interval_text](#) [16][3]
Provides the entries for the Interval dropdown menu in the Pattern Editor window.
- const char [c_chord_text](#) [8][5]
Provides the entries for the Chord dropdown menu in the Pattern Editor window.
- const int [c_max_instruments](#)
Provides the maximum number of instruments that can be defined in the ~/.seq24usr or ~/.config/sequencer64/sequencer.rc file.
- const int [c_max_busses](#)
Provides the maximum number of MIDI buss definitions supported in the "user" file.
- static const std::string [versiontext](#)
Sets up the "hardwired" version text for Sequencer64.
- static struct option [long_options](#) []
A structure for command parsing that provides the long forms of command-line arguments, and associates them with their respective short form.
- static const std::string [s_arg_list](#)
Provides a complete list of the short options, and is passed to getopt_long().

- static `lash * s_global_lash_driver`
The global pointer to the LASH driver instance.
- static const int `c_status_replace`
Purely internal constants used with the functions that implement MIDI control for the application.
- const int `c_mainwid_x`
The width of the main pattern/sequence grid, in pixels.
- static const int `c_select_all_notes`
Actions.

11.1.1 Detailed Description

0.9.3 delta-tick calculation code. This code doesn't quite work for generating the proper rate of MIDI clocks, and so have disabled that code until we can figure out what it is we're doing wrong. Do not enable it unless you are willing to test it.

11.1.2 Typedef Documentation

11.1.2.1 typedef long seq64::midipulse

HOWEVER, CURRENTLY, if you make this value unsigned, then perfrroll won't show any notes in the sequence bars!!!

11.1.3 Enumeration Type Documentation

11.1.3.1 enum seq64::seq_modifier_t

We have to tweak the names to avoid redeclaration errors and to "personalize" the values. We change "GDK" to "SEQ64".

Since we're getting events from, say Gtk-2.4, but using our (matching) values for comparison, use the `CAST_EQ↔UIVALENT()` macro to compare them. Note that we might still end up having to a remapping (e.g. if trying to get the code to work with the Qt framework).

Enumerator

SEQ64_SUPER_MASK Bits 13 and 14 are used by XKB, bits 15 to 25 are unused. Bit 29 is used internally.

11.1.3.2 enum seq64::seq_event_type_t

Only the values we need have been grabbed. We have to tweak the names to avoid redeclaration errors and to "personalize" the values. We change "GDK" to "SEQ64", but, for convenience (to hide errors? :-D), we keep the number the same.

Since we're getting events from, say Gtk-2.4, but using our (matching) values for comparison, use the `CAST_EQ↔UIVALENT()` macro to compare them. Note that we might still end up having to a remapping (e.g. if trying to get the code to work with the Qt framework).

11.1.3.3 enum seq64::seq_scroll_direction_t

We have to tweak the names to avoid redeclaration errors and to "personalize" the values. We change "SEQ64" to "SEQ64".

Since we're getting events from, say Gtk-2.4, but using our (matching) values for comparison, use the `CAST_EQUIVALENT()` macro to compare them. Note that we might still end up having to a remapping (e.g. if trying to get the code to work with the Qt framework).

11.1.3.4 enum seq64::clock_e

Enumerator

e_clock_off A clock enumeration. Not sure yet what these mean.

This enumeration was also defined in `midibus_portmidi.h`, but we put it into this common module to avoid duplication.

Corresponds to the "Off" selection in the MIDI Clock tab. With this setting, the MIDI Clock is disabled for the buss using this setting. Notes will still be sent that buss, of course. Some software synthesizer might require this setting in order to make a sound.

e_clock_pos Corresponds to the "Pos" selection in the MIDI Clock tab. With this setting, MIDI Clock will be sent to this buss, and, if playback is starting beyond tick 0, then MIDI Song Position and MIDI Continue will also be sent on this buss.

e_clock_mod Corresponds to the "Mod" selection in the MIDI Clock tab. With this setting, MIDI Clock and MIDI Start will be sent. But clocking won't begin until the Song Position has reached the start modulo (in 1/16th notes) that is specified.

11.1.3.5 enum seq64::interaction_method_t

Moved here from the `globals.h` module.

11.1.3.6 enum seq64::c_music_scales

Scales can be shown in the piano roll as gray bars for reference purposes.

We've added three more scales; there are still a number of them that could be fruitfully added to the list of scales.

It would be good to offload this stuff into a new "scale" class.

11.1.3.7 enum seq64::draw_type

Enumerator

DRAW_FIN Provides a set of methods for drawing certain items.

11.1.4 Function Documentation

11.1.4.1 `bool seq64::extract_timing_numbers (const std::string & s, std::string & part_1, std::string & part_2, std::string & part_3, std::string & fraction)`

- measures : beats : divisions
 - "213:4:920"
 - "0:1:0"
- hours : minutes : seconds . fraction
 - "2:04:12.14"
 - "0:1:2"

Warning

This is not the most efficient implementation you'll ever see. At some point we will tighten it up. This function is tested in the seq64-tests project, in the "calculations_unit_test" module.

Returns

Returns true if a reasonable portion (3 numbers) was good for extraction. The fraction part will start with a period for easier conversion to fractional seconds.

11.1.4.2 `std::string seq64::pulses_to_string (midipulse p)`

Todo Still needs to be unit tested.

11.1.4.3 `std::string seq64::pulses_to_measurestring (midipulse p, const midi_timing & seqparms)`

Parameters

<i>p</i>	The number of MIDI pulses (clocks, divisions, ticks, you name it) to be converted. If the value is SEQ64_ILLEGAL_PULSE, it is converted to 0, because callers don't generally worry about such niceties, and the least we can do is convert illegal measure-strings (like "000:0:000") to a legal value.
<i>seqparms</i>	This small structure provides the beats/measure, beat-width, and PPQN that hold for the sequence involved in this calculation.

Returns

Returns the absolute pulses that mark this duration.

11.1.4.4 `bool seq64::pulses_to_midi_measures (midipulse p, const midi_timing & seqparms, midi_measures & measures)`

$$m = p * W / (4 * P * B)$$

Parameters

<i>p</i>	Provides the MIDI pulses (as in "pulses per quarter note") that are to be converted to MIDI measures format.
<i>seqparms</i>	This small structure provides the beats/measure (B), beat-width (W), and PPQN (P) that hold for the sequence involved in this calculation. The beats/minute (T for tempo) value is not needed.
<i>measures</i>	Provides the current MIDI song time structure holding the results, which are the measures, beats, and divisions values for the time of interest. Note that the measures and beats are corrected to be re 1, not 0.

Returns

Returns true if the calculations were able to be made. The P, B, and W values all need to be greater than 0.

11.1.4.5 `std::string seq64::pulses_to_timestring (midipulse p, int bpm, int ppqn)`

If the fraction part is 0, then it is not shown. Examples:

```
- "0:0:0"
- "0:0:0.102333"
- "12:3:1"
- "12:3:1.000001"
```

Parameters

<i>p</i>	Provides the number of ticks, pulses, or divisions in the MIDI event time.
<i>bpm</i>	Provides the tempo of the song, in beats/minute.
<i>ppqn</i>	Provides the pulse-per-quarter-note of the song.

11.1.4.6 `std::string seq64::pulses_to_timestring (midipulse p, const midi_timing & timinginfo)`

See the [pulses_to_timestring\(\)](#) overload.

Todo Still needs to be unit tested.

Parameters

<i>p</i>	Provides the number of ticks, pulses, or divisions in the MIDI event time.
<i>timinginfo</i>	Provides the tempo of the song, in beats/minute, and the pulse-per-quarter-note of the song.

11.1.4.7 `midipulse seq64::measurestring_to_pulses (const std::string & measures, const midi_timing & seqparms)`

Parameters

<i>measures</i>	Provides the current MIDI song time in "measures:beats:divisions" format, where divisions are the MIDI pulses in "pulses-per-quarter-note".
<i>seqparms</i>	This small structure provides the beats/measure, beat-width, and PPQN that hold for the sequence involved in this calculation.

Returns

Returns the absolute pulses that mark this duration.

11.1.4.8 midipulse seq64::midi_measures_to_pulses (const midi_measures & *measures*, const midi_timing & *seqparms*)

$p = 4 * P * m * B / W$ p == pulse count (ticks or pulses) m == number of measures B == beats per measure (constant) P == pulses per quarter-note (constant) W == beat width in beats per measure (constant)

Note that the 0-pulse MIDI measure is "1:1:0", which means "at the beginning of the first beat of the first measure, no pulses". It is not "0:0:0" as one might expect.

Parameters

<i>measures</i>	Provides the current MIDI song time structure holding the measures, beats, and divisions values for the time of interest.
<i>seqparms</i>	This small structure provides the beats/measure, beat-width, and PPQN that hold for the sequence involved in this calculation.

Returns

Returns the absolute pulses that mark this duration. If the pulse-value cannot be calculated, then SEQ64_ILLEGAL_PULSE is returned.

11.1.4.9 midipulse seq64::timestring_to_pulses (const std::string & *timestring*, int *bpm*, int *ppqn*)

Parameters

<i>timestring</i>	The time value to be converted, which must be of the form "hh:mm:ss" or "hh:mm:ss.fraction".
<i>bpm</i>	The beats-per-minute tempo (e.g. 120) of the current MIDI song.
<i>ppqn</i>	The parts-per-quarter note precision (e.g. 192) of the current MIDI song.

Returns

Returns 0 if an error occurred or if the number actually translated to 0.

11.1.4.10 midipulse seq64::string_to_pulses (const std::string & *s*, const midi_timing & *mt*)

First, the type of string is deduced by the characters in the string. If the string contains two colons and a decimal point, it is assumed to be a time-string ("hh:mm:ss.frac"); in addition ss will have to be less than 60.

If the string just contains two colons, then it is assumed to be a measure-string ("measures:beats:divisions").

If it has none of the above, it is assumed to be pulses. Testing is not rigorous.

Parameters

<i>s</i>	Provides the string to convert to pulses.
<i>mt</i>	Provides the structures needed to provide BPM and other values needed for some of the conversions.

Returns

Returns the string as converted to MIDI pulses (divisions, clocks, ticks, whatever you call it).

11.1.4.11 `midibyte seq64::string_to_midibyte (const std::string & s)`

This function bypasses characters until it finds a digit (whether part of the number or a "0x" construct), and then converts it.

11.1.4.12 `std::string seq64::shorten_file_spec (const std::string & fpath, int leng)`

This is done by removing character in the middle, if necessary, and replacing them with an ellipse.

This function operates by first trying to find the `/home` directory. If found, it strips off `/home/username` and replace it with the Linux `~` replacement for the `$HOME` environment variable. This function assumes that the "username" portion *must* exist, and that there's no goofy stuff like double-slashes in the path.

Parameters

<i>fpath</i>	The file specification, including the full path to the file, and the name of the file.
<i>leng</i>	Provides the length to which to limit the string.

Returns

Returns the `fpath` parameter, possibly shortened to fit within the desired length.

11.1.4.13 `bool seq64::string_not_void (const std::string & s)`

Provides essentially the opposite test that [string_is_void\(\)](#) provides.

Parameters

<i>s</i>	The string pointer to check for emptiness.
----------	--

Returns

Returns 'true' if the pointer is valid, the string has a non-zero length, and is not just white-space.

The definition of white-space is provided by the `std::isspace()` function/macro.

11.1.4.14 `bool seq64::string_is_void (const std::string & s)`

Meant to have essentially the opposite result of `string_not_void()`. The meaning of empty is special here, as it refers to a string being useless as a token:

- The string is of zero length.
- The string has only white-space characters in it, where the `isspace()` macro provides the definition of white-space.

Parameters

<code>s</code>	The string pointer to check for emptiness.
----------------	--

Returns

Returns 'true' if the string has a zero length, or is only white-space.

11.1.4.15 `bool seq64::strings_match (const std::string & target, const std::string & x)`

The `strings_match()` function returns true if the comparison items are identical, without case-sensitivity in character content up to the length of the secondary string. This allows abbreviations to match. (And, in scanning routines, the first match is immediately accepted.)

Parameters

<i>target</i>	The primary string in the comparison. This is the target string, the one we hope to match. It is <i>assumed</i> to be non-empty, and the result is false if it is empty..
<i>x</i>	The secondary string in the comparison. It must be no longer than the target string, or the match is false.

Returns

Returns true if both strings are identical in characters, up to the length of the secondary string, with the case of the characters being insignificant. Otherwise, false is returned.

11.1.4.16 `int seq64::log2_time_sig_value (int tsd)`

Useful in converting a time signature's denominator to a Time Signature meta event's "dd" value.

Parameters

<i>tsd</i>	The time signature denominator, which must be a power of 2: 2, 4, 8, 16, or 32.
------------	---

Returns

Returns the power of 2 that achieves the *tsd* parameter value.

11.1.4.17 `void seq64::tempo_to_bytes (midibyte t[3], int tempo_us)`

Recall the format of a Tempo event:

0 FF 51 03 t2 t1 t0 (tempo as number of microseconds per quarter note)

This code is the inverse of the lines of code around line 768 in midifile.cpp, which is basically $((t2 * 256) + t1) * 256 + t0$.

As a test case, note that the default tempo is 120 beats/minute, which is equivalent to *ttttt*=500000 (0x07A120).

Parameters

<i>t</i>	Provides a small array of at least 3 elements to hold each tempo byte.
<i>tempo_us</i>	Provides the temp value in microseconds per quarter note.

11.1.4.18 `double seq64::beats_per_minute_from_tempo (double tempo) [inline]`

The tempo event's numeric value is given in 3 bytes, and is in units of microseconds-per-quarter-note (us/qn).

Parameters

<i>tempo</i>	The value of the Tempo meta-event, in units of us/qn. If this value is 0, we'll get an arithmetic exception.
--------------	--

Returns

Returns the beats per minute. If the tempo value is too small, then this function will crash. :-D

11.1.4.19 `double seq64::tempo_from_beats_per_minute (double bpm) [inline]`

Parameters

<i>bpm</i>	The value of beats-per-minute. If this value is 0, we'll get an arithmetic exception.
------------	---

Returns

Returns the tempo in qn/us. If the bpm value is too small, then this function will crash. :-D

11.1.4.20 `double seq64::pulse_length_us (int bpm, int ppqn) [inline]`

The formula for the pulse-length in seconds is:

$$P = \frac{60}{BPM * PPQN}$$

Parameters

<i>bpm</i>	Provides the beats-per-minute value. No sanity check is made. If this value is 0, we'll get an arithmetic exception.
<i>ppqn</i>	Provides the pulses-per-quarter-note value. No sanity check is made. If this value is 0, we'll get an arithmetic exception.

Returns

Returns the pulse length in microseconds. If either parameter is invalid, then this function will crash. :-D

11.1.4.21 `double seq64::delta_time_us_to_ticks (unsigned long us, int bpm, int ppqn) [inline]`

This function is the inverse of [ticks_to_delta_time_us\(\)](#).

Please note that terms "ticks" and "pulses" are equivalent, and refer to the "pulses" in "pulses per quarter note".

$$P = 120 \frac{\text{beats}}{\text{minute}} * 192 \frac{\text{pulses}}{\text{beats}} * T \text{ us} * \frac{1 \text{ minute}}{60 \text{ sec}} * \frac{1 \text{ sec}}{1,000,000 \text{ us}}$$

Note that this formula assumes that a beat is a quarter note. If a beat is an eighth note, then the P value would be halved, because there would be only 96 pulses per beat. We will implement an additional function to account for the beat; the current function merely blesses some calculations made in the application.

Parameters

<i>us</i>	The number of microseconds in the delta time.
<i>bpm</i>	Provides the beats-per-minute value, otherwise known as the "tempo".
<i>ppqn</i>	Provides the pulses-per-quarter-note value, otherwise known as the "division".

Returns

Returns the tick value.

11.1.4.22 `double seq64::ticks_to_delta_time_us (midipulse delta_ticks, int bpm, int ppqn) [inline]`

The inverse of [delta_time_us_to_ticks\(\)](#).

Please note that terms "ticks" and "pulses" are equivalent, and refer to the "pulses" in "pulses per quarter note".

Old: `60000000.0 * double(delta_ticks) / (double(bpm) * double(ppqn));`

Parameters

<i>delta_ticks</i>	The number of ticks or "clocks".
<i>bpm</i>	Provides the beats-per-minute value, otherwise known as the "tempo".
<i>ppqn</i>	Provides the pulses-per-quarter-note value, otherwise known as the "division".

Returns

Returns the time value in microseconds.

11.1.4.23 `double seq64::clock_tick_duration_bogus (int bpm, int ppqn) [inline]`

Deprecated This is a somewhat bogus calculation used only for "statistical" output in the old perform module. Name changed to reflect this unfortunate fact. Use [pulse_length_us\(\)](#) instead.

$$us = \frac{60000000 \text{ ppqn}}{MIDI_CLOCK_IN_PPQN * bpm * ppqn}$$

MIDI_CLOCK_IN_PPQN is 24.

Parameters

<i>bpm</i>	Provides the beats-per-minute value. No sanity check is made. If this value is 0, we'll get an arithmetic exception.
<i>ppqn</i>	Provides the pulses-per-quarter-note value. No sanity check is made. If this value is 0, we'll get an arithmetic exception.

Returns

Returns the clock tick duration in microseconds. If either parameter is invalid, this will crash. Who wants to waste time on value checks here? :-D

11.1.4.24 `int seq64::clock_ticks_from_ppqn (int ppqn) [inline]`

Parameters

<i>ppqn</i>	The number of pulses per quarter note. For example, the default value for Seq24 is 192.
-------------	---

Returns

The integer value of $ppqn / 24$ [MIDI_CLOCK_IN_PPQN] is returned.

11.1.4.25 `double seq64::double_ticks_from_ppqn (int ppqn) [inline]`

The same as [clock_ticks_from_ppqn\(\)](#), but returned as a double float.

Parameters

<i>ppqn</i>	The number of pulses per quarter note.
-------------	--

Returns

The double value of `ppqn / 24 [SEQ64_MIDI_CLOCK_IN_PPQN]` is returned.

11.1.4.26 `midipulse seq64::measures_to_ticks (int bpm, int ppqn, int bw, int measures = 1) [inline]`

This function is called in `seqedit::apply_length()`, when the user selects a sequence length in measures. That function calculates the length in ticks. The number of pulses is given by the number of quarter notes times the pulses per quarter note. The number of quarter notes is given by the measures times the quarter notes per measure. The quarter notes per measure is given by the beats per measure times 4 divided by beat_width beats. So:

```
p = 4 * P * m * B / W
p == pulse count (ticks or pulses)
m == number of measures
B == beats per measure (constant)
P == pulses per quarter-note (constant)
W == beat width in beats per measure (constant)
```

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 `p = 100 * 4 * 4 * 192 / 4 = 76800` ticks. Seems small.

Parameters

<i>bpm</i>	The B value in the equation, beats/measure.
<i>ppqn</i>	The P value in the equation, pulses/qn.
<i>bw</i>	The W value in the equation, the denominator of the time signature. If this value is 0, we'll get an arithmetic exception (crash), so we just return 0 in this case..
<i>measures</i>	The M value in the equation. It defaults to 1, in case one desires a simple "ticks per measure" number.

Returns

Returns the L value (ticks or pulses) as calculated via the given equation. If `bw` is 0, then 0 is returned.

11.1.4.27 `bool seq64::help_check (int argc, char * argv[])`

Also check for the `-legacy` option. Finally, it also checks for the `"?"` option that people sometimes use as a guess to get help.

Parameters

<i>argc</i>	The number of command-line arguments.
<i>argv</i>	The array of command-line argument pointers.

Returns

Returns true only if -V, --version, -h, --help, or "?" were encountered. If the legacy options occurred, then rc().legacy_format(true) is called, as a side effect, because it will be needed before we parse the options.

11.1.4.28 bool seq64::parse_options_files (perform & p, int argc, char * argv[])

It probably requires this call preceding: Gtk::Main kit(argc, argv), to strip any GTK+-specific parameters the knowledgeable user may have added. Usage:

```
Gtk::Main kit(argc, argv);
seq64::gui_assistant_gtk2 gui;
seq64::perform p(gui);
```

It also requires the caller to call rc().set_defaults() and usr().set_defaults(). The caller can then use the command-line to make any modifications to the setting that will be used here. The biggest example is the -r/--reveal-alsa-ports option, which determines if the MIDI buss definition strings are read from the 'user' configuration file.

Instead of the legacy Seq24 names, we use the new configuration file-names, located in the ~/.config/sequencer64 directory. However, if they aren't found, we no longer fall back to the legacy configuration file-names. If the --legacy option is in force, use only the legacy configuration file-name. The code also ensures the directory exists. CURRENTLY LINUX-SPECIFIC. See the [rc_settings](#) class for how this works.

```
std::string cfg_dir = seq64::rc().home_config_directory();
if (cfg_dir.empty())
    return EXIT_FAILURE;
```

Change Note ca 2016-04-03 We were parsing the user-file first, but we now need to parse the rc-file first, to get the manual-alsa-ports option, so that we can avoid overriding the port names that the ALSA system provides, if the manual-alsa-option is false.

Parameters

<i>p</i>	Provides the perform object that will be affected by the new parameters.
<i>argc</i>	The number of command-line arguments.
<i>argv</i>	The array of command-line argument pointers.

Returns

Returns true if the reading of both configuration files succeeded.

11.1.4.29 int seq64::parse_command_line_options (int argc, char * argv[])**Parameters**

<i>argc</i>	The number of command-line arguments.
<i>argv</i>	The array of command-line argument pointers.

Returns

Returns the value of `optind` if no help-related options were provided.

11.1.4.30 `bool seq64::write_options_files (const perform & p)`

This function gets any legacy global variables, on the theory that they might have been changed.

Returns

Returns true if both files were saved successfully. Even if one write failed, the other might have succeeded.

11.1.4.31 `std::string seq64::to_string (const event & ev)`

Nothing fancy. If you want that, use the `midicvt` project.

11.1.4.32 `bool seq64::file_exists (const std::string & filename)`**Parameters**

<i>filename</i>	Provides the name of the file to be checked.
-----------------	--

Returns

Returns 'true' if the file exists.

11.1.4.33 `bool seq64::file_readable (const std::string & filename)`**Parameters**

<i>filename</i>	Provides the name of the file to be checked.
-----------------	--

Returns

Returns 'true' if the file is readable.

11.1.4.34 `bool seq64::file_writable (const std::string & filename)`**Parameters**

<i>filename</i>	Provides the name of the file to be checked.
-----------------	--

Returns

Returns 'true' if the file is writable.

11.1.4.35 `bool seq64::file_accessible (const std::string & filename)`

An even stronger test than `file_exists`. At present, we see no need to distinguish read and write permissions. We assume the file is accessible only if the file has both permissions.

Parameters

<i>filename</i>	Provides the name of the file to be checked.
-----------------	--

Returns

Returns 'true' if the file is readable and writable.

11.1.4.36 `bool seq64::file_executable (const std::string & filename)`**Parameters**

<i>filename</i>	Provides the name of the file to be checked.
-----------------	--

Returns

Returns 'true' if the file exists.

11.1.4.37 `bool seq64::file_is_directory (const std::string & filename)`

This function is also used in the function of the same name in `fileutilities.cpp`.

Parameters

<i>filename</i>	Provides the name of the directory to be checked.
-----------------	---

Returns

Returns 'true' if the file is a directory.

11.1.4.38 `bool seq64::make_directory (const std::string & pathname)`

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

Parameters

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

Returns

Returns true if the path-name exists.

11.1.4.39 `int seq64::choose_ppqn (int ppqn)` `[inline]`

Putting it here means we can reduce the reliance on the global ppqn.

Parameters

<i>ppqn</i>	Provides the PPQN value to be used.
-------------	-------------------------------------

Returns

Returns the ppqn parameter, unless that parameter is SEQ64_USE_DEFAULT_PPQN (-1), then `usr().midi_↵_ppqn` is returned.

11.1.4.40 `bool seq64::ppqn_is_valid (int ppqn)` `[inline]`

Validates a PPQN value.

Parameters

<i>ppqn</i>	Provides the PPQN value to be used.
-------------	-------------------------------------

Returns

Returns true if the ppqn parameter is between MINIMUM_PPQN and MAXIMUM_PPQN, or is set to SE↵Q64_USE_DEFAULT_PPQN (-1).

11.1.4.41 `int seq64::jack_sync_callback (jack_transport_state_t state, jack_position_t * pos, void * arg)`

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

The transport state will be:

- `JackTransportStopped` when a new position is requested.
- `JackTransportStarting` when the transport is waiting to start.
- `JackTransportRolling` when the timeout has expired, and the position is now a moving target.

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.

Returns

Returns 1 if the function works, and 0 if something was wrong.

11.1.4.42 void seq64::jack_shutdown_callback (void * arg)

Parameters

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

11.1.4.43 void seq64::jack_timebase_callback (jack_transport_state_t state, jack_nframes_t nframes, jack_position_t * pos, int new_pos, void * arg)

The original version of the function worked properly with Hydrogen, but not with Klick. The new code seems to work with both. More testing and clarification is needed. This new code was "discovered" in the source-code for the "SooperLooper" project:

<http://essey.net/sooperlooper/>

The first difference with the new code is that it handles the case where the JACK position is moved (`new_pos == true`). If this is true, and the JackPositionBBT bit is off in `pos->valid`, then the new BBT value is set.

The seconds set of differences are in the "else" clause. In the new code, it is very simple: calculate the new tick value, back it off by the number of ticks in a beat, and perhaps go to the first beat of the next bar.

In the old code (complex!), the simple BBT adjustment is always made. This changes (perhaps) the `beats_per_bar`, `beat_type`, etc. We need to make these settings use the actual global values for beats set for Sequencer64. Then, if transitioning from JackTransportStarting to JackTransportRolling (instead of checking `new_pos!`), the BBT values (bar, beat, and tick) are finally adjusted. Here are the steps, with old and new steps noted:

```
-# Calculate the "delta" ticks based on the current frame, the
  ticks_per_beat, the beats_per_minute, and the frame_rate. The old
  code saves this in a local, the new code assigns it to pos->tick.
-# Old code: save this delta as a positive value.
-# Figure out the settings and modify bar, beat, tick, and
  bar_start_tick. The old and new code seem to have the same intent,
  but it seems like the new code is faster and also correct.
  - Old code: Calculations are made by division and mod
    operations.
  - New code: Calculations are made by increments and decrements
    in a while loop.
```

Parameters

<i>state</i>	Indicates the current state of JACK transport.
<i>nframes</i>	The number of JACK frames in the current time period.
<i>pos</i>	Provides the position structure to be filled in, the address of the position structure for the next cycle; <code>pos->frame</code> will be its frame number. If <code>new_pos</code> is FALSE, this structure contains extended position information from the current cycle. If TRUE, it contains whatever was set by the requester. The <code>timebase_callback</code> 's task is to update the extended information here.
<i>new_pos</i>	TRUE (non-zero) for a newly requested pos, or for the first cycle after the <code>timebase_callback</code> is defined. This is usually 0 in Sequencer64 at present, and 1 if one, say, presses "rewind" in <code>qjackctl</code> .
<i>arg</i>	Provides the jack_assistant pointer, currently unchecked for nullity.

11.1.4.44 `void seq64::jack_session_callback (jack_session_event_t * ev, void * arg)`

Glib is then used to connect in `perform::jack_session_event()`. However, the `perform` object's GUI-support interface is used instead of the following, so that the `libseq64` library can be independent of a specific GUI framework:

```
Glib::signal_idle().
    connect(sigc::mem_fun(*jack, &jack_assistant::session_event));
```

Parameters

<i>ev</i>	The JACK event to be set.
<i>arg</i>	The pointer to the jack_assistant object. Currently not checked for nullity.

11.1.4.45 `bool seq64::create_lash_driver (perform & p, int argc, char ** argv)`

Initializes the lash driver (strips lash-specific command line arguments), then connects to the LASH daemon and polls events.

This function will always be called from the main routine, and called only once. Note that we don't need that darn `SEQ64_LASH_SUPPORT` macro in client code anymore.

Parameters

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

Returns

This function returns true if a lash object was created. This function will not create one in not configured to, if the command-line options did not specify the creation of the LASH driver, or if the LASH driver was already created.

11.1.4.46 `lash * seq64::lash_driver ()`

Returns

Returns the pointer to the LASH driver if it exists. Otherwise a null pointer is returned. The caller *must always check* the return value.

11.1.4.47 `void seq64::delete_lash_driver ()`

This function will always be called from the main routine, once. The other lash-pointer functions will know if the pointer has been deleted.

11.1.4.48 `void * seq64::output_thread_func (void * myperf)`

Set up the performance, set the process to realtime privileges, and then start the output function.

Parameters

<i>myperf</i>	Provides the perform object instance that is to be used. Its output_func() is called.
---------------	---

11.1.4.49 `long seq64::min (long a, long b)` `[inline]`

Parameters

<i>a</i>	First operand.
<i>b</i>	Second operand.

11.1.4.50 `font& seq64::font_render ()` `[inline]`

We've going to render this pointer obsolete, though, and use a smart factory function to ensure the existence of this pointer, and return a reference to the font object.

We wanted to make the font a const object, but `mainwid::on_realize()` calls the `font::init()` function with its window object, and using const is impractical. We don't want to force every caller to deal with the overhead of passing even a null window pointer, either.

However, at some point we need some guarantee that the `init()` function is called before rendering a string. Right now, we guarantee it only by build order.

11.1.4.51 `Gtk::Adjustment & seq64::adjustment_dummy ()`

This static object is used so we have an Adjustment to assign to the Adjustment members for classes that don't use them. Clumsy? We shall see.

Anyway, the parameters for this constructor are value, lower, upper, step-increment, and two more values.

11.1.5 Variable Documentation

11.1.5.1 `std::string seq64::c_controller_names`

This array is used only by the seqedit class.

11.1.5.2 `const midibyte seq64::EVENT_NOTE_OFF`

The following MIDI events are channel messages. The comments represent the one or two data-bytes of the message.

Note that Channel Mode Messages use the same code as the Control Change, but uses reserved controller numbers ranging from 122 to 127.

11.1.5.3 `const midibyte seq64::EVENT_MIDI_SYSEX`

The following MIDI events have no channel. We have included redundant constant variables for the SysEx Start and End bytes just to make it clear that they are part of this sequence of values, though usually treated separately.

Only the following constants are followed by some data bytes:

```
- EVENT_MIDI_SYSEX          = 0xF0
- EVENT_MIDI_QUARTER_FRAME  = 0xF1      // undefined?
- EVENT_MIDI_SONG_POS       = 0xF2
- EVENT_MIDI_SONG_SELECT    = 0xF3
```

11.1.5.4 `const midibyte seq64::EVENT_NULL_CHANNEL`

However, it also means that the channel is encoded in the `m_status` byte itself. This is our work around to be able to hold a multi-channel SMF 0 track in a sequence. In a Sequencer64 SMF 0 track, every event has a channel. In a Sequencer64 SMF 1 track, the events do not have a channel. Instead, the channel is a global value of the sequence, and is stuffed into each event when the event is played or is written to a MIDI file.

11.1.5.5 `const int seq64::c_midibus_output_size`

These constants were also defined in `midibus_portmidi.h`, but we made them common to both implementations here.

11.1.5.6 `const midilong seq64::c_midibus`

Some of the information is stored with each track (and in the `midi_container`-derived classes), and some is stored in the proprietary header.

Track (sequencer-specific) data:

```
c_midibus
c_midich
c_timesig
c_triggers
c_triggers_new
c_musickey
c_musicscale
```

Footer ("proprietary") data:

```
c_midictrl
c_midiclocks
c_notes
c_bpmtag (beats per minute)
c_mutegroups
c_backsequence
```

Also see the PDF file in the following project for more information about the "proprietary" data:

<https://github.com/ahlstromcj/sequencer64-doc.git>

Note that the track data is read from the MIDI file, but not written directly to the MIDI file. Instead, it is stored in the MIDI container as sequences are edited to used these "sequencer-specific" features. Also note that `c_triggers` has been replaced by `c_triggers_new` as the code that marks the triggers stored with a sequence.

As an extension, we will eventually grab the last key, scale, and background sequence value selected in a sequence and write them as track data, where they can be read in and applied to a specific sequence, when the `seqedit` object is created. These values would not be stored in the legacy format.

Something like this could be done in the "user" configuration file, but then the key and scale would apply to all songs. We don't want that.

We could also add snap and note-length to the per-song defaults, but the "user" configuration file seems like a better place to store these preferences.

11.1.5.7 `const int seq64::c_midi_track_ctrl`

The lowest value is `c_seqs_in_set * 2 = 64`.

I think the reason for that value is to perhaps handle two sets or something like that. Will figure it out later.

The controls are read in from the "rc" configuration files, and are written to the `c_midictrl` section of the "proprietary" final track in a Seq24/Sequencer64 MIDI file.

11.1.5.8 `const bool seq64::c_scales_policy[c_scale_size][SEQ64_OCTAVE_SIZE]`

See the following sites for more information:

<http://method-behind-the-music.com/theory/scalesandkeys/>

https://en.wikipedia.org/wiki/Heptatonic_scale

Note that melodic minor descends in the same way as the natural minor scale, so it descends differently than it ascends. We don't deal with that trick, at all.

Chromatic	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	Notes, chord
Major	C	.	D	.	E	F	.	G	.	A	.	B	
Minor	C	.	D	Eb	.	F	.	G	Ab	.	Bb	.	
Harmonic Minor	C	.	D	Eb	.	F	.	G	Ab	.	.	B	
Melodic Minor	C	.	D	Eb	.	F	.	G	.	A	.	B	Descending diff.
C Whole Tone	C	.	D	.	E	.	F#	.	G#	.	A#	.	C+7 chord
C Lydian Dominant	C	.	D	.	E	.	F#	.	G	.	A	Bb	Unimplemented, A7
A Mixolydian	C#	.	D	.	E	.	F#	.	G	.	A	.	Unimplemented, A7
B Whole Tone	.	Db	.	Eb	.	F	.	G	.	A	.	B	Unimplemented
G Whole Tone	.	C#	.	D#	.	F	.	G	.	A	.	B	Unimplemented, same
G Octatonic	.	C#	D	.	E	F	.	G	Ab	.	Bb	B	Unimplemented

11.1.5.9 `const int seq64::c_scales_transpose_up[c_scale_size][SEQ64_OCTAVE_SIZE]`

For example, if we simply add 1 semitone to each note, it remains a minor key, but it is in a different minor key. Using the transpositions in these arrays, the minor key remains the same minor key.

Major	C . D . E F . G . A . B
Transpose up	2 0 2 0 1 2 0 2 0 2 0 1
Result up	D . E . F G . A . B . C
Minor	C . D D# . F . G G# . A# .
Transpose up	2 0 1 2 0 2 0 1 2 0 2 0
Result up	D . D# F . G . G# A# . C .
Harmonic minor	C . D Eb . F . G Ab . . B
Transpose up	2 . 1 2 . 2 . 1 3 . . 1
Result up	D . Eb F . G . Ab B . . C
Melodic minor	C . D Eb . F . G . A . B
Transpose up	2 . 1 2 . 2 . 2 . 2 . 1
Result up	D . Eb F . G . A . B . C
C Whole Tone	C . D . E . F# . G# . A# .
Transpose up	2 . 2 . 2 . 2 . 2 . 2 .
Result up	D . E . F# . G# . A# . C .

11.1.5.10 `const int seq64::c_scales_transpose_dn_neg[c_scale_size][SEQ64_OCTAVE_SIZE]`

Just remember to negate each value before using it.

Major	C . D . E F . G . A . B
Transpose down	1 0 2 0 2 1 0 2 0 2 0 2
Result down	B . C . D E . F . G . A
Minor	C . D D# . F . G G# . A# .
Transpose down	2 0 2 1 0 2 0 2 1 0 2 0
Result down	A# . C D . D# . F G . G# .
Harmonic minor	C . D Eb . F . G Ab . . B
Transpose down	1 . 2 1 . 2 . 2 1 . . 3
Result down	B . C D . Eb . F G . . Ab
Melodic minor	C . D Eb . F . G . A . B
Transpose down	1 . 2 1 . 2 . 2 . 2 . 2
Result down	B . C D . Eb . F . G . A
C whole tone	C . D . E . F# . G# . A# .
Transpose down	2 . 2 . 2 . 2 . 2 . 2 .
Result down	A# . C . D . E . F# . G# .

11.1.5.11 `const char seq64::c_chord_text[8][5]`

However, I have not seen this menu in the GUI! Ah, it only appears if the user has selected a musical scale like Major or Minor.

11.1.5.12 `const int seq64::c_max_instruments`

With a value of 64, this is more of a sanity-check than a realistic number of instruments defined by a user.

11.1.5.13 `const std::string seq64::versiontext` `[static]`

This value ultimately comes from the `configure.ac` script.

11.1.5.14 `struct option seq64::long_options[]` `[static]`

Note the terminating null structure..

11.1.5.15 `lash* seq64::s_global_lash_driver` `[static]`

It is actually hidden in this module now, so that a function can be used in its place.

Like the font renderer, This item was once created in the main module, `sequencer64.cpp`. Now we make it a safer, more fool-proof, function. However, unlike the font-render, which always exists, the LASH driver is conditional, and might not be wanted. Therefore, we cannot return a reference, because there's no such thing as a null reference in C++. We have to return a pointer.

11.1.5.16 `const int seq64::c_mainwid_x`

Affected by the `c_mainwid_border` and `c_mainwid_spacing` values.

11.1.5.17 `const int seq64::c_select_all_notes` `[static]`

These variables represent actions that can be applied to a selection of notes. One idea would be to add a swing-quantize action. We will reserve the value here, for notes only; not yet used or part of the action menu.

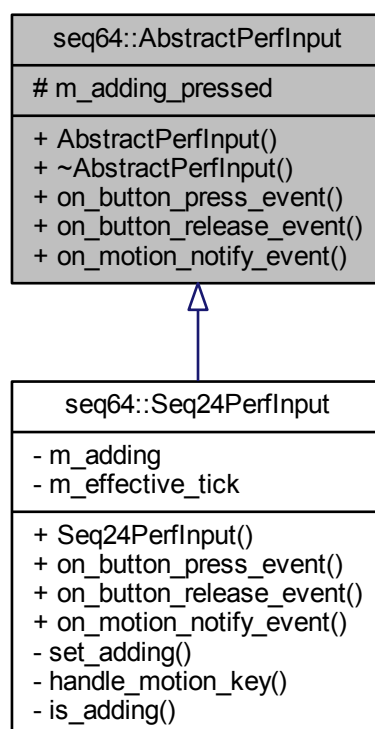
Chapter 12

Data Structure Documentation

12.1 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:



12.2 seq64::automutex Class Reference

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

Public Member Functions

- [automutex](#) ([mutex](#) &my_mutex)
Principal constructor gets a reference to a mutex parameter, and then locks the mutex.
- [~automutex](#) ()
The destructor unlocks the mutex.

Private Attributes

- [mutex](#) & [m_safety_mutex](#)
Provides the mutex reference to be used for locking.

12.2.1 Detailed Description

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

12.3 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.

12.3.1 Detailed Description

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

12.3.2 Constructor & Destructor Documentation

12.3.2.1 seq64::click::click ()

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

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

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

Parameters

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

12.3.2.3 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.
------------	--

12.3.3 Member Function Documentation

12.3.3.1 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.
------------	--

Returns

Returns a reference to self for usage in a string of assignments.

12.3.4 Field Documentation

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

0 is the left-most coordinate.

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

0 is the top-most coordinate.

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

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

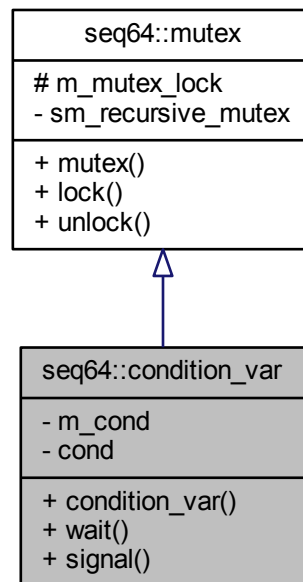
12.3.4.4 seq_modifier_t seq64::click::m_modifier [private]

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

12.4 seq64::condition_var Class Reference

A mutex works best in conjunction with a condition variable.

Inheritance diagram for seq64::condition_var:



Public Member Functions

- `condition_var ()`
Initialize the condition variable with the global variable.
- `void wait ()`
Waits for the condition variable.
- `void signal ()`
Signals the condition variable.

Private Attributes

- `pthread_cond_t m_cond`
Provides a class-specific condition variable.

Static Private Attributes

- `static const pthread_cond_t cond`
Provides a "global" condition variable.

Additional Inherited Members

12.4.1 Detailed Description

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

12.4.2 Field Documentation

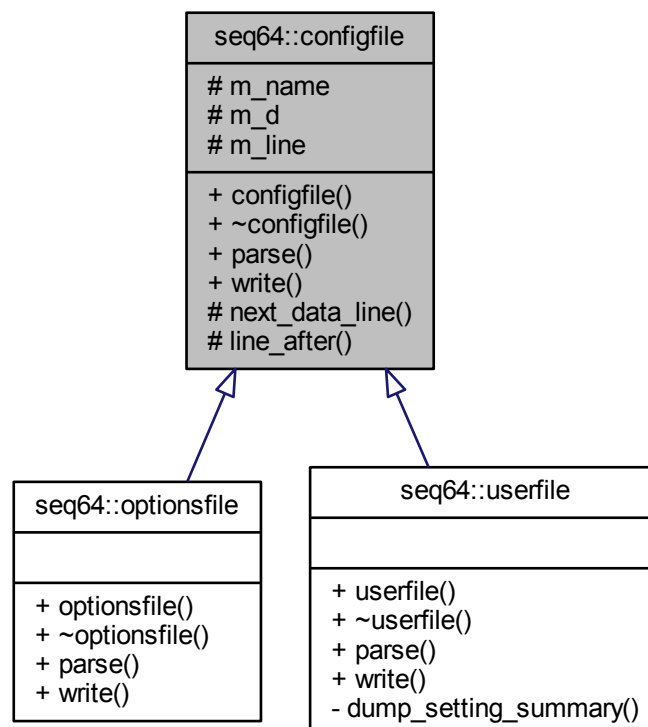
12.4.2.1 `const pthread_cond_t seq64::condition_var::cond` `[static], [private]`

Define the static condition variable used by all mutex locks.

12.5 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 &name)
Provides the string constructor for a configuration file.
- virtual [~configfile](#) ()
A rote destructor needed for a base class.

Protected Member Functions

- bool [next_data_line](#) (std::ifstream &file)
Gets the next line of data from an input stream.
- void [line_after](#) (std::ifstream &file, const std::string &tag)
This function gets a specific line of text, specified as a tag.

Protected Attributes

- std::string [m_name](#)
Provides the name of the configuration 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.

12.5.1 Constructor & Destructor Documentation

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

Parameters

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

12.5.2 Member Function Documentation

12.5.2.1 bool 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.

Member m_line is a "global" return value.

Parameters

<i>file</i>	Points to an input stream. We converted this item to a reference; pointers can be subject to problems. For example, what if someone passes a null pointer?
-------------	--

Returns

Returns true if a presumed data line was found. False is returned if not found before an EOF or a section marker ("[" is found. This is a new (ca 2016-02-14) feature of this function, to assist in adding new data to the file.

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

Then it gets the next non-blank line (i.e. data line) after that.

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. Normally, the tag is a section marker, such as "[user-interface]". Best to assume an exact match is needed.

12.5.3 Field Documentation

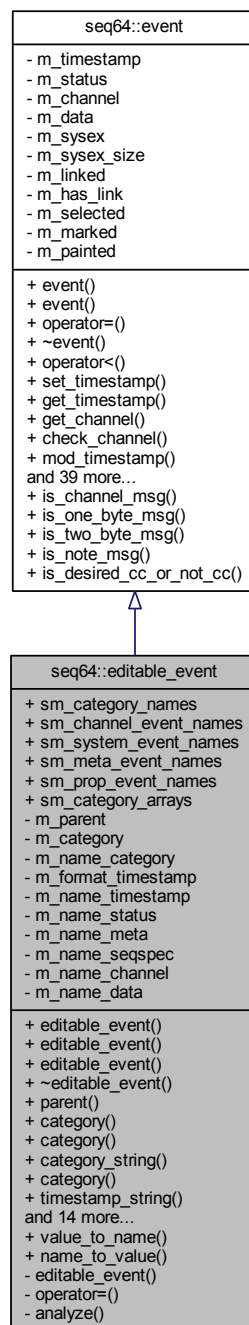
12.5.3.1 `char seq64::configfile::m_line[SEQ64_LINE_MAX]` `[protected]`

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

12.6 seq64::editable_event Class Reference

Provides for the management of MIDI editable events.

Inheritance diagram for seq64::editable_event:



Public Types

Public Member Functions

- `editable_event` (const `editable_events` &`parent`)

This constructor simply initializes all of the class members.

- `editable_event` (const `editable_events` &`parent`, const `event` &`ev`)
Event constructor.
- `editable_event` (const `editable_event` &`rhs`)
This copy constructor initializes most of the class members.
- virtual `~editable_event` ()
This destructor current is a rote virtual function override.
- const `editable_events` & `parent` () const
'Getter' function for member `m_parent`
- `category_t category` () const
'Getter' function for member `m_category`
- void `category` (`category_t` `c`)
'Setter' function for member `m_category` by value Also keeps the `m_name_category` member in synchrony.
- const std::string & `category_string` () const
'Getter' function for member `m_category`
- void `category` (const std::string &`cs`)
'Setter' function for member `m_category` by value Also keeps the `m_name_category` member in synchrony, but looks up the name, rather than using the name parameter, to avoid storing abbreviations.
- const std::string & `timestamp_string` () const
'Getter' function for member `m_name_timestamp`
- `midipulse timestamp` () const
'Getter' function for member `event::get_timestamp()` Implemented to allow a uniform naming convention that is not slavish to the get/set crowd [this ain't Java].
- void `timestamp` (`midipulse` `ts`)
'Setter' function for member `event::set_timestamp()` Implemented to allow a uniform naming convention that is not slavish to the get/set crowd [this ain't Java].
- void `timestamp` (const std::string &`ts_string`)
'Setter' function for member `event::set_timestamp()` [string version]
- std::string `time_as_pulses` ()
Converts the current time-stamp to a string representation in units of pulses.
- std::string `time_as_measures` ()
Converts the current time-stamp to a string representation in units of measures, beats, and divisions.
- std::string `time_as_minutes` ()
Converts the current time-stamp to a string representation in units of hours, minutes, seconds, and fraction.
- void `set_status_from_string` (const std::string &`ts`, const std::string &`s`, const std::string &`sd0`, const std::string &`sd1`)
Converts a string into an event status, along with timestamp and data bytes.
- std::string `format_timestamp` ()
Formats the current timestamp member as a string.
- std::string `stock_event_string` ()
Converts the event into a string describing the full event.
- std::string `status_string` () const
'Getter' function for member `m_name_status`
- std::string `meta_string` () const
'Getter' function for member `m_name_meta`
- std::string `seqspec_string` () const
'Getter' function for member `m_name_seqspec`
- std::string `channel_string` () const
'Getter' function for member `m_name_channel`
- std::string `data_string` () const
'Getter' function for member `m_name_data`

Static Public Member Functions

- static std::string [value_to_name](#) (midibyte value, [category_t](#) cat)
Provides a static lookup function that returns the name, if any, associated with a midibyte value.
- static unsigned short [name_to_value](#) (const std::string &name, [category_t](#) cat)
Provides a static lookup function that returns the value, if any, associated with a name string.

Static Public Attributes

- static const name_value_t [sm_category_names](#) []
An array of event categories and their names.
- static const name_value_t [sm_channel_event_names](#) []
An array of MIDI channel events and their names.
- static const name_value_t [sm_system_event_names](#) []
An array of MIDI system events and their names.
- static const name_value_t [sm_meta_event_names](#) []
An array of Meta events and their names.
- static const name_value_t [sm_prop_event_names](#) []
An array of Sequencer64-specific events and their names.
- static const name_value_t *const [sm_category_arrays](#) []
Provides for fast access (no ifs) to the correct name array for the given category.

Private Member Functions

- void [analyze](#) ()
Analyzes an editable-event to make all the settings it needs.

Private Attributes

- const [editable_events](#) & [m_parent](#)
Provides a reference to the container that holds this event.
- [category_t](#) [m_category](#)
Indicates the overall category of this event, which will be [category_channel_message](#), [category_system_message](#), [category_meta_event](#), and [category_prop_event](#).
- std::string [m_name_category](#)
Holds the name of the event category for this event.
- [timestamp_format_t](#) [m_format_timestamp](#)
Indicates the format to display the time-stamp.
- std::string [m_name_timestamp](#)
Holds the string version of the MIDI pulses time-stamp.
- std::string [m_name_status](#)
Holds the name of the status value for this event.
- std::string [m_name_meta](#)
Holds the name of the meta message, if applicable.
- std::string [m_name_seqspec](#)
If we eventually implement the editing of the Seq24/Sequencer64 "proprietary" meta sequencer-specific events, the name of the SeqSpec will be stored here.
- std::string [m_name_channel](#)
Holds the channel description, if applicable.
- std::string [m_name_data](#)
Holds the data description, if applicable.

12.6.1 Detailed Description

It makes the following members of an event modifiable using human-readable strings:

```
- m_timestamp
- m_status
- m_channel
- m_data[]
```

Eventually, it would be nice to be able to edit, or at least view, the SysEx events and the Meta events. Those two will require extensions to make events out of them (SysEx is partly supported).

To the concepts of event, the `editable_event` class adds a category field and strings to represent all of these members.

12.6.2 Member Enumeration Documentation

12.6.2.1 `enum seq64::editable_event::category_t`

Enumerator

category_name These values determine the major kind of event, which determines what types of events are possible for this editable event object. These tags are accompanied by category names in `sm_category_names[]`. The enum values are cast to midibyte values for the purposes of using the lookup infrastructure.

Indicates that the lookup needs to be done on the category names, as listed in `sm_category_names[]`.

category_channel_message Indicates a channel event, with a value ranging from 0x80 through 0xEF. Some examples are note on/off, control change, and program change. Values are looked up in `sm_channel_event_names[]`.

category_system_message Indicates a system event, with a value ranging from 0xF0 through 0xFF. Some examples are SysEx start/end, song position, and stop/start/continue/reset. Values are looked up in `sm_system_event_names[]`.

category_meta_event Indicates a meta event, and there is a second value that is used to look up the name of the meta event, in `sm_meta_event_names[]`.

category_prop_event Indicates a "proprietary", Sequencer64 event. Indicates to look up the name of the event in `sm_prop_event_names[]`. Not sure if these kinds of events will be stored separately.

12.6.2.2 `enum seq64::editable_event::timestamp_format_t`

Enumerator

timestamp_measures Provides a code to indicate the desired timestamp format. Three are supported. All editable events will share the same timestamp format, but it seems good to make this a event class member, rather than something imposed from an outside static value. We shall see.

This format displays the time in "measures:beats:divisions" format, where measures and beats start at 1. Thus, "1:1:0" is equivalent to 0 pulses or to "0:0:0.0" in normal time values.

timestamp_time This format displays the time in "hh:mm:second.fraction" format. The value displayed should not depend upon the internal timing parameters of the event.

timestamp_pulses This format specifies a bare pulse format for the timestamp – a long integer ranging from 0 on up. Obviously, this representation depends on the PPQN value for the sequence holding this event.

12.6.3 Constructor & Destructor Documentation

12.6.3.1 seq64::editable_event::editable_event (const editable_events & parent)

editable_event::editable_event () : event (), m_category (category_name), m_name_category (), m_format_↵ timestamp (timestamp_measures), m_name_timestamp (), m_name_status (), m_name_meta (), m_name_↵ seqspec (), m_name_channel (), m_name_data () { // Empty body } Principal constructor.

12.6.3.2 seq64::editable_event::editable_event (const editable_events & parent, const event & ev)

This function basically adds all of the extra [editable_event](#) stuff to a standard event, so that the resulting [editable_↵ _event](#) is container-ready.

12.6.3.3 seq64::editable_event::editable_event (const editable_event & rhs)

This function is currently geared only toward support of the SMF 0 channel-splitting feature. Many of the members are not set to useful values when the MIDI file is read, so we don't handle them for now.

Warning

This function does not yet copy the SysEx data. The inclusion of SysEx [editable_events](#) was not complete in Seq24, and it is still not complete in Sequencer64. Nor does it currently bother with the links.

Parameters

<i>rhs</i>	Provides the editable_event object to be copied.
------------	--

12.6.4 Member Function Documentation

12.6.4.1 std::string seq64::editable_event::value_to_name (midibyte value, editable_event::category_t cat) [static]

Parameters

<i>value</i>	The MIDI byte value to look up.
<i>cat</i>	The category of the MIDI byte. Each category calls a different name array into play.

Returns

Returns the name associated with the value. If there is no such name, then an empty string is returned.

12.6.4.2 unsigned short seq64::editable_event::name_to_value (const std::string & name, editable_event::category_t cat) [static]

The string_match() function, which can match abbreviations, case-insensitively, is used to make the string comparisons.

Parameters

<i>name</i>	The string value to look up.
<i>cat</i>	The category of the MIDI byte. Each category calls a different name array into play.

Returns

Returns the value associated with the name. If there is no such value, then SEQ64_END_OF_MIDIBYTE_↔TABLE is returned.

12.6.4.3 void seq64::editable_event::category (category_t c)

Note that a bad value is translated to the value of category_name.

12.6.4.4 void seq64::editable_event::category (const std::string & name)

Note that a bad value is translated to the value of category_name.

12.6.4.5 void seq64::editable_event::timestamp (midipulse ts)

Plus, we also have to set the string version at the same time.

Parameters

<i>ts</i>	Provides the timestamp in units of MIDI pulses.
-----------	---

The format of the string representation is of the format selected by the m_format_timestamp member and is set by the [format_timestamp\(\)](#) function.

12.6.4.6 void seq64::editable_event::timestamp (const std::string & ts_string)

Parameters

<i>ts_string</i>	Provides the timestamp in units of MIDI pulses.
------------------	---

The format of the string representation is of the format selected by the m_format_timestamp member and is set by the [format_timestamp\(\)](#) function.

12.6.4.7 std::string seq64::editable_event::time_as_measures ()

Cannot be inlined because of a circular dependency between the [editable_event](#) and [editable_events](#) classes.

12.6.4.8 `std::string seq64::editable_event::time_as_minutes ()`

Cannot be inlined because of a circular dependency between the [editable_event](#) and [editable_events](#) classes.

12.6.4.9 `void seq64::editable_event::set_status_from_string (const std::string & ts, const std::string & s, const std::string & sd0, const std::string & sd1)`

Currently, this function handles only the following two messages:

- `category_channel_message`
- `category_system_message`

After all of the numbering member items have been set, they are converted and assigned to the string versions via a call to the [analyze\(\)](#) function.

Parameters

<i>ts</i>	Provides the time-stamp string of the event.
<i>s</i>	Provides the name of the event, such as "Program Change".
<i>sd0</i>	Provides the string defining the first data byte of the event.
<i>sd1</i>	Provides the string defining the second data byte of the event, if applicable to the event.

12.6.4.10 `std::string seq64::editable_event::format_timestamp ()`

The format of the string representation is of the format selected by the `m_format_timestamp` member.

12.6.4.11 `std::string seq64::editable_event::stock_event_string ()`

We get the time-stamp as a string, make sure the event is fully analyzed so that all items and strings are set correctly.

12.6.4.12 `void seq64::editable_event::analyze () [private]`

Used in the constructors. Some of the setters indirectly set the appropriate string representation, as well.

Category:

This function can figure out if the status byte implies a channel message or a system message, and set the category string as well. However, at this time, detection of Meta events (0xFF) or Proprietary/SeqSpec events (0xFF with 0x2424) aren't able to be detected due to lack of context here (and due to the fact that currently such events are not yet stored in a sequence, and the least-significant-byte gets masked off anyway.)

Status:

We distinguish between channel and system messages, and then one- and two-byte messages, but don't yet distinguish the data values fully.

12.6.5 Field Documentation

12.6.5.1 `const editable_event::name_value_t seq64::editable_event::sm_category_names` `[static]`

Initializes the array of event/name pairs for the MIDI events categories.

Terminated by an empty string, the latter being the preferred test, for consistency with the other arrays and because 0 is often a legitimate code value.

12.6.5.2 `const editable_event::name_value_t seq64::editable_event::sm_channel_event_names` `[static]`

Initializes the array of event/name pairs for the channel MIDI events.

We split channel and system messages into two arrays, for semantic reasons and for faster linear lookups.

Terminated by an empty string.

12.6.5.3 `const editable_event::name_value_t seq64::editable_event::sm_system_event_names` `[static]`

Initializes the array of event/name pairs for the system MIDI events.

We split channel and system messages into two arrays, for semantic reasons and for faster linear lookups.

Terminated by an empty string.

12.6.5.4 `const editable_event::name_value_t seq64::editable_event::sm_meta_event_names` `[static]`

Initializes the array of event/name pairs for all of the Meta events.

Terminated only by the empty string.

12.6.5.5 `const editable_event::name_value_t seq64::editable_event::sm_prop_event_names` `[static]`

Initializes the array of event/name pairs for all of the seq24/sequencer64-specific events.

Terminated only by the empty string. Note that the numbers reflect the masking off of the high-order bits by 0x242400FF.

12.6.5.6 `const editable_event::name_value_t *const seq64::editable_event::sm_category_arrays` `[static]`

Contains pointers (references cannot be stored in an array) to the desired array for a given category.

Too bad that an array of references is not possible.

This code could be considered a bit rococo.

12.6.5.7 `const editable_events& seq64::editable_event::m_parent` [private]

The container's "children" need to go to their "parent" to get certain items of information.

12.6.5.8 `category_t seq64::editable_event::m_category` [private]

The category_name value is not set here, since that category is used only for looking up the human-readable form of the category.

12.6.5.9 `timestamp_format_t seq64::editable_event::m_format_timestamp` [private]

The default is to display in timestamp_measures format.

12.6.5.10 `std::string seq64::editable_event::m_name_status` [private]

It will include the names of the channel messages and the system messages. The latter includes SysEx and Meta messages.

12.6.5.11 `std::string seq64::editable_event::m_name_meta` [private]

If not applicable, this name will be empty.

12.7 seq64::editable_events Class Reference

Provides for the management of an ordered collection MIDI editable events.

Public Member Functions

- `editable_events` (sequence &seq, int bpm)
This constructor hooks into the sequence object.
- `editable_events` (const `editable_events` &rhs)
This copy constructor initializes most of the class members.
- `editable_events & operator=` (const `editable_events` &rhs)
This principal assignment operator sets most of the class members.
- virtual `~editable_events` ()
This destructor current is a rote virtual function override.
- const `midi_timing & timing` () const
'Getter' function for member m_midi_parameters
- `midipulse string_to_pulses` (const std::string &ts_string) const
Calculates the MIDI pulses (divisions) from a string using one of the free functions of the calculations module.
- bool `load_events` ()
Accesses the sequence's event-list, iterating through it from beginning to end, wrapping each event in the list in an editable event and inserting it into the editable-event container.
- bool `save_events` ()

Erases the sequence's event container and recreates it using the edited container of editable events.

- Events & `events` ()
'Getter' function for member `m_events`
- 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`.
- bool `add` (const `event` &`e`)
Adds an event, converted to an `editable_event`, to the internal event list.
- bool `add` (const `editable_event` &`e`)
Adds an editable event to the internal event list.
- bool `replace` (iterator `ie`, const `editable_event` &`e`)
Provides a wrapper for the iterator form of `erase()`, which is the only one that the `editable_events` container uses.
- 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()`.
- iterator `current_event` () const
'Getter' function for member `m_current_event` The caller must make sure the iterator is not `Events::end()`.

Private Types

- typedef `event_list::event_key` `Key`
Types to use to with the multimap implementation.

Private Member Functions

- void `current_event` (iterator `cei`)
'Setter' function for member `m_current_event`

Private Attributes

- Events `m_events`
Holds the `editable_events`.
- iterator `m_current_event`
Points to the current event, which is the event that has just been inserted.
- `sequence` & `m_sequence`
Provides a reference to the sequence containing the events to be edited.
- `midi_timing` `m_midi_parameters`
Holds the current settings for the sequence (and usually for the whole MIDI tune as well).

12.7.1 Member Typedef Documentation

12.7.1.1 `typedef event_list::event_key seq64::editable_events::Key` `[private]`

These typenames are identical to those used in [event_list](#), but of course they are in the [editable_events](#) scope instead.

12.7.2 Constructor & Destructor Documentation

12.7.2.1 `seq64::editable_events::editable_events (sequence & seq, int bpm)`

Parameters

<i>seq</i>	Provides a reference to the sequence object, which provides the events and some of the MIDI timing parameters.
<i>bpm</i>	Provides the beats/minute value, which the caller figures out how to get and provides in this parameter.

12.7.2.2 `seq64::editable_events::editable_events (const editable_events & rhs)`

Note that we need to reconstitute the event links here, as well.

Parameters

<i>rhs</i>	Provides the editable_events object to be copied.
------------	---

12.7.3 Member Function Documentation

12.7.3.1 `editable_events & seq64::editable_events::operator= (const editable_events & rhs)`

Note that we need to reconstitute the event links here, as well.

Parameters

<i>rhs</i>	Provides the editable_events object to be assigned.
------------	---

Returns

Returns a reference to "this" object, to support the serial assignment of [editable_eventss](#).

12.7.3.2 `bool seq64::editable_events::load_events ()`

Note that the new events will not have valid links (actually, no links). These links are used for associating Note Off events with their respective Note On events. To be consistent, we must take the time to reconstitute these links, using [event_list::verify_and_link\(\)](#).

Returns

Returns true if the size of the final `editable_event` container matches the size of the original events container.

12.7.3.3 `bool seq64::editable_events::save_events ()`

Note that the old events are replaced only if the container of editable events is not empty. There are safer ways for the user to erase all the events.

Todo Consider what to do about the `sequence::m_is_modified` flag.

Returns

Returns true if the size of the final event container matches the size of the original `editable_events` container.

12.7.3.4 `int seq64::editable_events::count () const` `[inline]`

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

12.7.3.5 `bool seq64::editable_events::add (const event & e)`**Parameters**

<code>e</code>	Provides the regular event to be added to the list of editable events.
----------------	--

Returns

Returns true if the insertion succeeded, as evidenced by an increment in container size.

12.7.3.6 `bool seq64::editable_events::add (const editable_event & e)`

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;
```

Parameters

<code>e</code>	Provides the regular event to be added to the list of editable events.
----------------	--

Returns

Returns true if the insertion succeeded, as evidenced by an increment in container size.

Side-effect(s) Sets `m_current_event`, which can be used right-away in a single-threaded context to get an iterator to the event via the `current_event()` accessor.

12.7.4 Field Documentation

12.7.4.1 iterator seq64::editable_events::m_current_event [private]

(From this event we can get the current time and other parameters.) If the container were a plain map, we could instead use a key to access it. But we can at least use an iterator, rather than a bare pointer.

12.7.4.2 sequence& seq64::editable_events::m_sequence [private]

Besides the events, this object also holds the beats/measure, beat-width, and the PPQN value. The beats/minute have to be obtained from the application's perform object, and passed to the [editable_events](#) constructor by the caller.

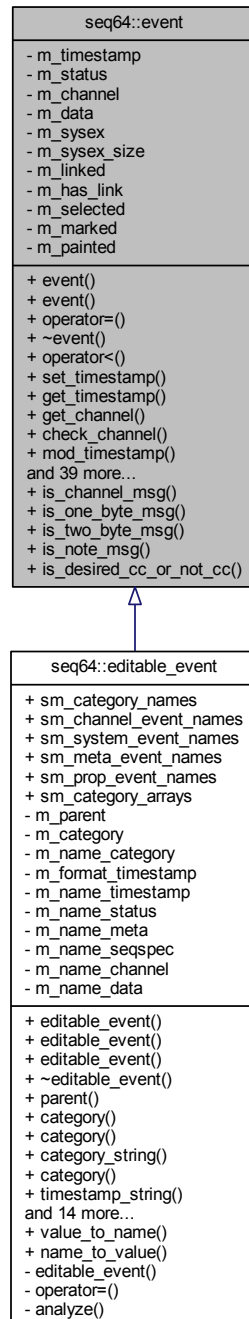
12.7.4.3 midi_timing seq64::editable_events::m_midi_parameters [private]

It holds the beats/minute, beats/measure, beat-width, and PPQN values needed to properly convert MIDI pulse timestamps to time and measure values.

12.8 seq64::event Class Reference

Provides events for management of MIDI events.

Inheritance diagram for seq64::event:



Public Member Functions

- `event ()`
This constructor simply initializes all of the class members.
- `event (const event &rhs)`
This copy constructor initializes most of the class members.
- `event & operator= (const event &rhs)`

- This principal assignment operator sets most of the class members.*

 - virtual `~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` (midipulse time)

'Setter' function for member `m_timestamp`
 - midipulse `get_timestamp` () const

'Getter' function for member `m_timestamp`
 - midibyte `get_channel` () const

'Getter' function for member `m_channel`
 - bool `check_channel` (int channel) const

Checks the channel number to see if the event's channel matches it, or if the event has no channel.
 - void `mod_timestamp` (midipulse 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 status.
 - void `set_status` (midibyte eventcode, midibyte channel)

This overload is useful when synthesizing events, such as converting a Note On event with a velocity of zero to a Note Off event.
 - void `set_channel` (midibyte channel)

Sets the channel "nybble", without modifying the status "nybble".
 - 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 `restart_sysex` ()

Deletes and clears out the SYSEX buffer.
 - bool `append_sysex` (midibyte *data, int len)

Appends SYSEX data to a new buffer.
 - midibyte * `get_sysex` () const

'Getter' function for member `m_sysex`
 - void `set_sysex_size` (int len)

'Setter' function for member `m_sysex_size`
 - int `get_sysex_size` () const

'Getter' function for member `m_sysex_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 ()` const
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 m)
Static test for channel messages/statuses.
- static bool `is_one_byte_msg` (midibyte m)
Static test for channel messages that have only one data byte.
- static bool `is_two_byte_msg` (midibyte m)
Static test for channel messages that have two data bytes.
- static bool `is_note_msg` (midibyte m)
Static test for messages that involve notes and velocity.
- static bool `is_desired_cc_or_not_cc` (midibyte m, 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

- midipulse `m_timestamp`
Provides the MIDI timestamp in ticks, otherwise known as the "pulses" in "pulses per quarter note" (PPQN).
- midibyte `m_status`
This is the status byte without the channel.
- midibyte `m_channel`
In order to be able to handle MIDI channel-splitting of an SMF 0 file, we need to store the channel, even if we override it when playing the MIDI data.
- 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.
- int `m_sysex_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.".

12.8.1 Detailed Description

A MIDI event consists of 3 bytes:

- # Status byte, lsssnnn, 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.

12.8.2 Constructor & Destructor Documentation

12.8.2.1 seq64::event::event (const event & rhs)

This function is currently geared only toward support of the SMF 0 channel-splitting feature. Many of the members are not set to useful values when the MIDI file is read, so we don't handle them for now.

Note that now events are also copied when creating the [editable_events](#) container, so this function is even more important. The event links, for linking Note Off events to their respective Note On events, are dropped. Generally, they will need to be reconstituted by calling the [event_list::verify_and_link\(\)](#) function.

Warning

This function does not yet copy the SysEx data. The inclusion of SysEx events was not complete in Seq24, and it is still not complete in Sequencer64. Nor does it currently bother with the links, as noted above.

Parameters

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

12.8.2.2 seq64::event::~~event () [virtual]

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

12.8.3 Member Function Documentation

12.8.3.1 event & seq64::event::operator= (const event & rhs)

This function is currently geared only toward support of the SMF 0 channel-splitting feature. Many of the member are not set to useful value when the MIDI file is read, so we don't handle them for now.

Warning

This function does not yet copy the SysEx data. The inclusion of SysEx events was not complete in Seq24, and it is still not complete in Sequencer64. Nor does it currently bother with the links.

Parameters

<i>rhs</i>	Provides the event object to be assigned.
------------	---

Returns

Returns a reference to "this" object, to support the serial assignment of events.

12.8.3.2 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::multimap` 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, but may cause the upper limit of the number of playing sequences to drop a little, due to the overhead of incrementing multimap iterators versus list iterators).

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.

12.8.3.3 `bool seq64::event::check_channel (int channel) const [inline]`

Used in the SMF 0 track-splitting code.

Parameters

<i>channel</i>	The channel to check.
----------------	-----------------------

Returns

Returns true if the given channel matches the event's channel.

12.8.3.4 `static bool seq64::event::is_channel_msg (midibyte m) [inline],[static]`

This function requires that the channel data have already been masked off.

Parameters

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

Returns

Returns true if the byte represents a MIDI channel message.

12.8.3.5 `static bool seq64::event::is_one_byte_msg (midibyte m) [inline],[static]`

The rest have two.

Parameters

<i>m</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.

12.8.3.6 `static bool seq64::event::is_two_byte_msg (midibyte m) [inline],[static]`

Parameters

<i>m</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.

12.8.3.7 `static bool seq64::event::is_note_msg (midibyte m) [inline],[static]`

Parameters

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

Returns

Returns true if the byte represents a MIDI note message.

12.8.3.8 `static bool seq64::event::is_desired_cc_or_not_cc (midibyte m, 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 m
    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.
```

12.8.3.9 void seq64::event::mod_timestamp (midipulse *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.

12.8.3.10 void seq64::event::set_status (midibyte *status*)

If *a_status* is a 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()`.

Parameters

<i>status</i>	The status byte, perhaps read from a MIDI file or edited in the sequencer's event editor. Sometime, this byte will have the channel nybble masked off. If that is the case, the eventcode/channel overload of this function is more appropriate.
---------------	--

12.8.3.11 void seq64::event::set_status (midibyte *eventcode*, midibyte *channel*)

Parameters

<i>eventcode</i>	The status byte, perhaps read from a MIDI file. This byte is assumed to have already had its low nybble cleared by masking against EVENT_CLEAR_CHAN_MASK.
<i>channel</i>	The channel byte. Combined with the event-code, this makes a valid MIDI "status" byte. This byte is assume to have already had its high nybble cleared by masking against EVENT_GET_CHAN_MASK.

12.8.3.12 `void seq64::event::set_channel (midibyte channel) [inline]`

Note that the sequence channel generally overrides this value.

Parameters

<i>channel</i>	The channel byte.
----------------	-------------------

12.8.3.13 `void seq64::event::set_data (midibyte d1) [inline]`

Parameters

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

12.8.3.14 `void seq64::event::set_data (midibyte d1, midibyte d2) [inline]`

Parameters

<i>d1</i>	The first byte value to set.
<i>d2</i>	The second byte value to set.

12.8.3.15 `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.

12.8.3.16 `bool seq64::event::append_sysex (midibyte * data, int dsize)`

First, a buffer of size `m_sysex_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.

12.8.3.17 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.

12.8.4 Field Documentation**12.8.4.1 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.

12.8.4.2 midibyte seq64::event::m_channel [private]

This member adds another 4 bytes to the event object, most likely.

12.8.4.3 midibyte seq64::event::m_data[SEQ64_MIDI_DATA_BYTE_COUNT] [private]

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

12.8.4.4 midibyte* seq64::event::m_sysex [private]

This really ought to be a Boost or STD scoped pointer. Currently, it doesn't seem to be used.

12.8.4.5 bool seq64::event::m_has_link [private]

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

12.9 seq64::event_list::event_key Class Reference

Provides a key value for an event map.

Public Member Functions

- [event_key](#) ([midipulse](#) 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

- [midipulse](#) m_timestamp
The primary key-value for the key.
- int [m_rank](#)
The sub-key-value for the key.

12.9.1 Detailed Description

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

12.9.2 Constructor & Destructor Documentation

12.9.2.1 seq64::event_list::event_key::event_key ([midipulse](#) tstamp, int rank)

Parameters

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

12.9.2.2 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.
------------	--------------------------------------

12.9.3 Member Function Documentation

12.9.3.1 bool seq64::event_list::event_key::operator< (const [event_key](#) & rhs) const

Parameters

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

Returns

Returns true if the rank and timestamp of the current object are less than those of rhs.

12.9.4 Field Documentation

12.9.4.1 midipulse seq64::event_list::event_key::m_timestamp [private]

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

12.10 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.
- bool [empty](#) () const
Returns true if there are no events.
- bool [add](#) (const [event](#) &e, bool postsort=true)
Adds an event to the internal event list in an optionally sorted manner.

- bool `is_modified ()` const
'Getter' function for member `m_is_modified`
- void `unmodify ()`
'Setter' function for member `m_is_modified` This function may be needed by some of the sequence editors.
- 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 (midipulse 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 (midipulse slength)`
Marks all events that have a time-stamp that is out of range.
- void `mark_all ()`
Marks all events.
- 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](#) () const
 - 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.*
- bool [m_is_modified](#)
 - A new flag to indicate if an event was added or removed.*

12.10.1 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.

12.10.2 Constructor & Destructor Documentation

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

Parameters

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

12.10.3 Member Function Documentation

12.10.3.1 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.
------------	---

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

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

12.10.3.3 `bool 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.

Returns

Returns true if the insertion succeeded, as evidenced by an increment in container size.

12.10.3.4 `void seq64::event_list::unmodify () [inline]`

But use it with great caution.

12.10.3.5 `void seq64::event_list::remove (iterator ie) [inline]`

Currently, no check on removal is performed. Set the modified-flag.

12.10.3.6 `void seq64::event_list::clear () [inline]`

Set the modified-flag.

12.10.3.7 `void seq64::event_list::merge (event_list & el, bool presort = true)`

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

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

The merge effectively removes all the elements in T (which becomes empty), and inserts them into their ordered position within container (which expands in size by the number of elements transferred). The operation is performed

without constructing nor destroying any element, whether T is an lvalue or an rvalue, or whether the value-type supports move-construction or not.

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

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

Parameters

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

12.10.3.8 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.

12.10.3.9 void seq64::event_list::verify_and_link (midipulse *slength*) [private]

Threadsafe

Parameters

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

12.10.3.10 void seq64::event_list::mark_out_of_range (midipulse *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.

Note

This code was comparing the timestamp as greater than or equal to the sequence length. However, being equal is fine. This may explain why the midifile code would add one tick to the length of the last note when processing the end-of-track.

Parameters

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

12.10.3.11 `void seq64::event_list::mark_all () [private]`

Not yet used, but might come in handy with the event editor dialog.

12.10.3.12 `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.

12.10.3.13 `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.

12.10.4 Field Documentation

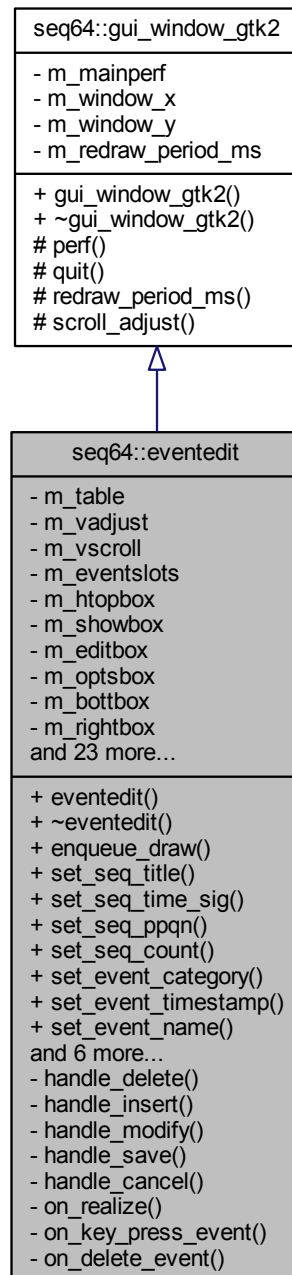
12.10.4.1 `bool seq64::event_list::m_is_modified [private]`

We may need to give client code a way to reload the sequence. This is currently an issue when a seqroll and an eventedit/eventslots are active for the same sequence.

12.11 seq64::eventedit Class Reference

This class supports an Event Editor that is used to tweak the details of events and get a better idea of the mix of events in a sequence.

Inheritance diagram for seq64::eventedit:



Public Member Functions

- [eventedit](#) ([perform](#) &p, [sequence](#) &seq)
Principal constructor, has a reference to a perform object.
- [~eventedit](#) ()
This rote constructor does nothing.
- void [enqueue_draw](#) ()

- Helper wrapper for calling `perfroll::queue_draw()` for one or both event edits.*

 - void `set_seq_title` (const std::string &title)
Sets `m_label_seq_name` to the title.
 - void `set_seq_time_sig` (const std::string &sig)
Sets `m_label_time_sig` to the time-signature string.
 - void `set_seq_ppqn` (const std::string &p)
Sets `m_label_ppqn` to the parts-per-quarter-note string.
 - void `set_seq_count` ()
Sets `m_label_ev_count` to the number-of-events string.
 - void `set_event_category` (const std::string &c)
Sets `m_label_category` to the category string.
 - void `set_event_timestamp` (const std::string &ts)
Sets `m_entry_ev_timestamp` to the time-stamp string.
 - void `set_event_name` (const std::string &n)
Sets `m_entry_ev_name` to the name-of-event string.
 - void `set_event_data_0` (const std::string &d)
Sets `m_entry_ev_data_0` to the first data byte string.
 - void `set_event_data_1` (const std::string &d)
Sets `m_entry_data_1` to the second data byte string.
 - void `perf_modify` ()
Provides a way to mark the perform object as modified, when the modified sequence is saved.
 - void `set_dirty` (bool flag=true)
Sets the "modified" status of the user-interface.
 - void `v_adjustment` (int value)
Sets the parameters for the vertical scroll-bar, using only the value parameter.
 - void `v_adjustment` (int value, int lower, int upper)
Sets the parameters for the vertical scroll-bar that is associated with the eventslots event-list user-interface.

Private Member Functions

- void `handle_delete` ()
Initiates the deletion of the current editable event.
- void `handle_insert` ()
Initiates the insertion of a new editable event.
- void `handle_modify` ()
Passes the edited fields to the current editable event in the eventslot.
- void `handle_save` ()
Handles saving the edited data back to the original sequence.
- void `handle_cancel` ()
Cancels the edits and closes the dialog box.
- void `on_realize` ()
This callback function calls the base-class `on_realize()` function.
- bool `on_key_press_event` (GdkEventKey *ev)
This function is the callback for a key-press event.
- bool `on_delete_event` (GdkEventAny *event)
Handles an on-delete event.

Private Attributes

- `Gtk::Table * m_table`
A whole horde of GUI elements.
- `Gtk::Label * m_label_index`
Items to size the m_indexslots member.
- `Gtk::Label * m_label_seq_name`
Items for the inside of the m_showbox member.
- `Gtk::Label * m_label_category`
Items for the inside of the m_editbox member.
- `Gtk::Label * m_label_right`
Padding for the right side of the user-interface.
- `sequence & m_seq`
A reference to the sequence being edited, to control its editing flag.

Additional Inherited Members

12.11.1 Constructor & Destructor Documentation

12.11.1.1 seq64::eventedit::eventedit (perform & p, sequence & seq)

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

Adjustment parameters:

<code>value</code>	initial value
<code>lower</code>	minimum value
<code>upper</code>	maximum value
<code>step_increment</code>	step increment
<code>page_increment</code>	page increment
<code>page_size</code>	page size

Table constructor parameters:

<code>rows</code>
<code>columns</code>
<code>homogenous</code>

Table attach() parameters:

<code>child</code>	widget to add.
<code>left_attach</code>	column number to attach left side of a child widget
<code>right_attach</code>	column number to attach right side of a child widget
<code>top_attach</code>	row number to attach the top of a child widget
<code>bottom_attach</code>	row number to attach the bottom of a child widget
<code>xoptions</code>	properties of the child widget when table resized
<code>yoptions</code>	same as xoptions, except vertical.
<code>xpadding</code>	padding on L and R of widget added to table
<code>ypadding</code>	amount of padding above and below the child widget

Layout:

We're going to change the layout.

```

0          1  2          3  4
-----
htop | (OLD LAYOUT)          :  :          |
----- showbox -----|
e'slots | 1-120:0:192 Program Change | ^ | "Sequence name" | |
-----| | 4/4 PPQN 192 | r |
| 2-120:1:0 Program Change | s | 9999 events | i | | | |
|-----| c |-----| g |
| ...      ...      ... | r | Channel Event: Ch. 5 | h |
| ...      ...      ... | o |-----| t |
| ...      ...      ... | l | [Edit field: Note On ] | |
| ...      ...      ... | l |-----| b |
| ...      ...      ... | | [Edit field: Key # ] | o |
| ...      ...      ... | b |-----| x |
| ...      ...      ... | a | [Edit field: Vel # ] | |
| ...      ...      ... | r |-----| |
| ...      ...      ... | | [Optional more data? ] | |
| ...      ...      ... | |-----| |
| ...      ...      ... | | o Pulses | |
| ...      ...      ... | | o Measures | |
| ...      ...      ... | v | o Time | |
|-----| |-----| |
| 56-136:3:133 Program Change | v | | Save | | Close | |
-----
14

```

Parameters

<i>p</i>	Refers to the main performance object.
<i>seq</i>	Refers to the sequence holding the event data to be edited.

12.11.1.2 seq64::eventedit::~~eventedit ()

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

12.11.2 Member Function Documentation

12.11.2.1 void seq64::eventedit::set_seq_title (const std::string & title)

Parameters

<i>title</i>	The name of the sequence.
--------------	---------------------------

12.11.2.2 void seq64::eventedit::set_seq_time_sig (const std::string & sig)

Parameters

<i>sig</i>	The time signature of the sequence.
------------	-------------------------------------

12.11.2.3 void seq64::eventedit::set_seq_ppqn (const std::string & p)

Parameters

<i>p</i>	The parts-per-quarter-note string for the sequence.
----------	---

12.11.2.4 void seq64::eventedit::set_event_category (const std::string & *c*)

Parameters

<i>c</i>	The category string for the current event.
----------	--

12.11.2.5 void seq64::eventedit::set_event_timestamp (const std::string & *ts*)

Parameters

<i>ts</i>	The time-stamp string for the current event.
-----------	--

12.11.2.6 void seq64::eventedit::set_event_name (const std::string & *n*)

Parameters

<i>n</i>	The name-of-event string for the current event.
----------	---

12.11.2.7 void seq64::eventedit::set_event_data_0 (const std::string & *d*)

Parameters

<i>d</i>	The first data byte string for the current event.
----------	---

12.11.2.8 void seq64::eventedit::set_event_data_1 (const std::string & *d*)

Parameters

<i>d</i>	The second data byte string for the current event.
----------	--

12.11.2.9 void seq64::eventedit::set_dirty (bool *flag* = true)

This includes changing a label and enabling/disabling the Save button.

Parameters

<i>flag</i>	If true, the modified status is indicated, otherwise it is cleared.
-------------	---

12.11.2.10 void seq64::eventedit::v_adjustment (int *value*)

This function overload provides a common use case.

Parameters

<i>value</i>	The new current value to be indicated by the scroll-bar.
--------------	--

12.11.2.11 void seq64::eventedit::v_adjustment (int value, int lower, int upper)

It keeps the frame scroll-bar in sync with the frame movement actions. Some of the parameters are obtained from the eventslots object:

- Page size comes from eventslots::line_maximum().
- Page increment is a little less than the page-size value.

Parameters

<i>value</i>	The current value to be indicated by the scroll-bar. It will lie between the lower and upper parameter.
<i>lower</i>	The lowest value to be indicated by the scroll-bar.
<i>upper</i>	The highest value to be indicated by the scroll-bar.

12.11.2.12 void seq64::eventedit::handle_insert () [private]

The event's location will be determined by the timestamp and existing events. Note that we have to recalibrate the scroll-bar when we insert/delete events by calling [v_adjustment\(\)](#).

12.11.2.13 void seq64::eventedit::handle_modify () [private]

Note that there are two cases to worry about. If the timestamp has not changed, then we can simply modify the existing current event in place. Otherwise, we need to delete the old event and insert the new one. But that is done for us by [eventslots::modify_current_event\(\)](#).

12.11.2.14 void seq64::eventedit::handle_save () [private]

The event list in the original sequence is cleared, and the editable events are converted to plain events, and added to the container, one by one.

Todo Could also support writing the events to a new sequence, for added flexibility.

12.11.2.15 bool seq64::eventedit::on_key_press_event (GdkEventKey * ev) [private]

If the Up or Down arrow is pressed (later, k and j :-), then we tell the eventslots object to move the "current event" highlighting up or down. In Gtkmm, these arrows also cause movement from one edit field to the next, so we disable that process if the event was handled here.

Note that some vi-like keys were supported, but they are needed for the edit fields, so cannot be used here. Also, the Delete key is needed for the edit fields. For now, we replace it with the asterisk, which is easy to access from the numeric pad of a keyboard, and allows for rapid deletion. The Insert key also causes confusing effects in the edit fields, so we replace it by the slash. Note that the asterisk and slash should not be required in any of the edit fields.

HOWEVER, there are still some issues with "/", so you'll just have to click the button to insert an event.

Parameters

<i>ev</i>	The key event to process.
-----------	---------------------------

Returns

Returns true if the event got handled somewhere along the line.

12.11.2.16 `bool seq64::eventedit::on_delete_event (GdkEventAny * event)` `[private]`

It sets the sequence object's editing flag to false, and deletes "this". This function is called if the "Close" ("X") button in the window's title bar is clicked. That is a different action from clicking the Close button.

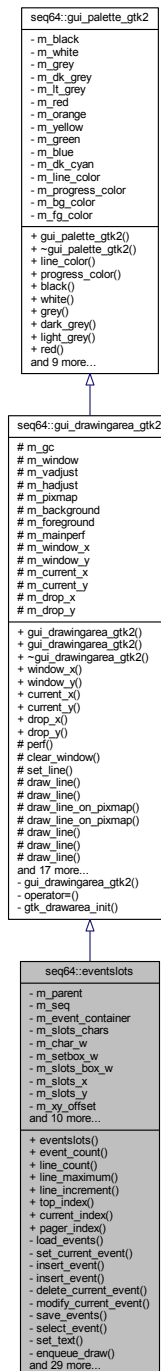
Returns

Always returns false.

12.12 seq64::eventslots Class Reference

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

Inheritance diagram for seq64::eventslots:



Public Member Functions

- `eventslots` (perform &p, `eventedit` &parent, `sequence` &seq, `Gtk::Adjustment` &vadjust)

Principal constructor for this user-interface object.

- `int event_count () const`

'Getter' function for member `m_event_count` Returns the number of total events in the sequence represented by the `eventslots` object.

- int [line_count](#) () const
'Getter' function for member m_line_count Returns the current number of rows (events) in the eventslots's display.
- int [line_maximum](#) () const
'Getter' function for member m_line_maximum Returns the maximum number of rows (events) in the eventslots's display.
- int [line_increment](#) () const
Provides the "page increment" or "line increment" of the frame, This value is the current line-maximum of the frame minus its overlap value.
- int [top_index](#) () const
'Getter' function for member m_top_index
- int [current_index](#) () const
'Getter' function for member m_current_index
- int [pager_index](#) () const
'Getter' function for member m_pager_index

Private Member Functions

- bool [load_events](#) ()
Grabs the event list from the sequence and uses it to fill the editable-event list.
- void [set_current_event](#) (const editable_events::iterator ei, int index, bool full_redraw=true)
Set the current event, which is the event that is highlighted.
- bool [insert_event](#) (const [editable_event](#) &edev)
Inserts an event.
- bool [insert_event](#) (const std::string &evtimestamp, const std::string &evname, const std::string &evdata0, const std::string &evdata1)
Inserts an event based on the setting provided, which the eventedit object gets from its Entry fields.
- bool [delete_current_event](#) ()
Deletes the current event, and makes adjustments due to that deletion.
- bool [modify_current_event](#) (const std::string &evtimestamp, const std::string &evname, const std::string &evdata0, const std::string &evdata1)
Modifies the data in the currently-selected event.
- bool [save_events](#) ()
Writes the events back to the sequence.
- void [select_event](#) (int event_index=SEQ64_NULL_EVENT_INDEX, bool full_redraw=true)
Selects and highlights the event that is located in the frame at the given event index.
- void [set_text](#) (const std::string &evcategory, const std::string &evtimestamp, const std::string &evname, const std::string &evdata0, const std::string &evdata1)
Sets the text in the parent dialog, eventedit.
- void [enqueue_draw](#) ()
Wraps queue_draw().
- int [convert_y](#) (int y)
Converts a y-value into an event index relative to 0 (the top of the eventslots window/pixmap) and returns it.
- void [draw_event](#) (editable_events::iterator ei, int index)
Draw the given slot/event.
- void [draw_events](#) ()
Draws all of the events in the current eventslots frame.
- void [change_vert](#) ()
Change the vertical offset of events.
- void [page_movement](#) (int new_value)
Adjusts the vertical position of the frame according to the given new scrollbar/vadjust value.
- void [page_topper](#) (editable_events::iterator newcurrent)

- Adjusts the vertical position of the frame according to the given new bottom iterator.*

 - int [decrement_top](#) ()
Decrements the top iterator, if possible.
 - int [increment_top](#) ()
Increments the top iterator, if possible.
 - int [decrement_current](#) ()
Decrements the current iterator, if possible.
 - int [increment_current](#) ()
Increments the current iterator, if possible.
 - int [decrement_bottom](#) ()
Decrements the bottom iterator, if possible.
 - int [increment_bottom](#) ()
Increments the bottom iterator, if possible.
 - void [update_pixmap](#) ()
This function does nothing.
 - void [draw_area](#) ()
This function does nothing.
 - void [redraw](#) ()
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.
 - bool [on_focus_in_event](#) (GdkEventFocus *ev)
This callback is an attempt to get keyboard focus into the eventslots pixmap area.
 - bool [on_focus_out_event](#) (GdkEventFocus *ev)
This callback handles an out-of-focus event by resetting the flag HAS_FOCUS.
 - bool [on_scroll_event](#) (GdkEventScroll *ev)
Handle the scrolling of the window.
 - void [on_size_allocate](#) (Gtk::Allocation &)
Handles a size-allocation event.
 - void [on_move_up](#) ()
Move to the previous event.
 - void [on_move_down](#) ()
Move to the next event.
 - void [on_frame_up](#) ()
Move to the previous frame.
 - void [on_frame_down](#) ()
Move to the next frame.
 - void [on_frame_home](#) ()
Move to the first frame.
 - void [on_frame_end](#) ()
Move to the last frame.

Private Attributes

- [eventedit](#) & [m_parent](#)
Provides a link to the `eventedit` that created this object.
- [sequence](#) & [m_seq](#)
Provides a reference to the sequence that this dialog is meant to view or modify.
- [editable_events](#) [m_event_container](#)
Holds the editable events for this sequence.
- [int](#) [m_slots_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_slots_box_w](#)
Provides the width of the "slot" box.
- [int](#) [m_slots_x](#)
Provides the width of the names box, which is the width of a character for 24 characters.
- [int](#) [m_slots_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.
- [int](#) [m_event_count](#)
The current number of events in the edited container.
- [int](#) [m_line_count](#)
Counts the number of displayed events, which depends on how many events there are (`m_event_count`) and the size of the event list (`m_line_maximum`).
- [int](#) [m_line_maximum](#)
Counts the maximum number of displayed events, which depends on the size of the event list (and thus the size of the dialog box for the event editor).
- [int](#) [m_line_overlap](#)
Provides a little overlap for paging through the frame.
- [int](#) [m_top_index](#)
The index of the event that is 0th in the visible list of events.
- [int](#) [m_current_index](#)
Indicates the index of the current event within the frame.
- [editable_events::iterator](#) [m_top_iterator](#)
Provides the top "pointer" to the start of the editable-events section that is being shown in the user-interface.
- [editable_events::iterator](#) [m_bottom_iterator](#)
Provides the bottom "pointer" to the end of the editable-events section that is being shown in the user-interface.
- [editable_events::iterator](#) [m_current_iterator](#)
Provides the "pointer" to the event currently in focus.
- [int](#) [m_pager_index](#)
Indicates the event index that matches the index value of the vertical pager.

Additional Inherited Members

12.12.1 Member Function Documentation

12.12.1.1 `bool seq64::eventslots::load_events () [private]`

Determines how many events can be shown in the GUI [later] and adjusts the top and bottom editable-event iterators to show the first page of events.

Returns

Returns true if the event iterators were able to be set up as valid.

12.12.1.2 `void seq64::eventslots::set_current_event (const editable_events::iterator ei, int index, bool full_redraw = true) [private]`

Note in the `sprintf()` calls that the first digit is part of the data byte, so that translation is easier.

Parameters

<i>ei</i>	The iterator that points to the event.
<i>index</i>	The index (re 0) of the event, starting at the top line of the frame. It is a frame index, not a container index.
<i>full_redraw</i>	If true (the default) does a full redraw of the frame. Otherwise, only the current event is drawn. Generally, the only time a single event (actually, two adjacent events) is convenient to draw is when using the arrow keys, where the speed of keystroke auto-repeats makes the full-frame update scrolling very flickery and disconcerting.

12.12.1.3 `bool seq64::eventslots::insert_event (const editable_event & edev) [private]`

What actually happens here depends if the new event is before the frame, within the frame, or after the frame, based on the timestamp.

If before the frame: To keep the previous events visible, we do not need to increment the iterators (insertion does not affect multimap iterators), but we do need to increment their indices. The contents shown in the frame should not change.

If at the frame top: The new timestamp equals the top timestamp. We don't know exactly where the new event goes in the multimap, but we do have a new event.

If at the frame bottom: TODO

If after the frame: No action needed if the bottom event is actually at the bottom of the frame. But if the frame is not yet filled, we need to increment the bottom iterator, and its index.

Note

Actually, it is far easier to just adjust all the counts and iterators and redraw the screen, as done by the [page_topper\(\)](#) function.

Parameters

<i>edev</i>	The event to insert, prebuilt.
-------------	--------------------------------

Returns

Returns true if the event was inserted.

12.12.1.4 `bool seq64::eventslots::insert_event (const std::string & evtimestamp, const std::string & evname, const std::string & evdata0, const std::string & evdata1) [private]`

It calls the other [insert_event\(\)](#) overload.

Note that we need to qualify the temporary event class object we create below, with the [seq64](#) namespace, otherwise the compiler thinks we're trying to access some Gtkmm thing.

Parameters

<i>evtimestamp</i>	The time-stamp of the new event, as obtained from the event-edit timestamp field.
<i>evname</i>	The type name (status name) of the new event, as obtained from the event-edit event-name field.
<i>evdata0</i>	The first data byte of the new event, as obtained from the event-edit data 1 field.
<i>evdata1</i>	The second data byte of the new event, as obtained from the event-edit data 2 field. Used only for two-parameter events.

Returns

Returns true if the event was inserted.

12.12.1.5 `bool seq64::eventslots::delete_current_event () [private]`

To delete the current event, this function moves the current iterator to the next event, deletes the previously-current iterator, adjusts the event count and the bottom iterator, and redraws the pixmap. The exact changes depend upon whether the deleted event was at the top of the visible frame, within the visible frame, or at the bottom the visible frame. Note that only visible events can be the current event, and thus get deleted.

```

Event Index
0
1
2           Top
3 <----- Top case: The new top iterator, index becomes 2
4
.
.           Inside of Visible Frame
.
43
44          Bottom
45 <----- Top case: The new bottom iterator, index becomes 44
46          Bottom case: Same result

```

Basically, when an event is deleted, the frame (delimited by the event-index members) stays in place, while the frame iterators move to the previous event. If the top of the frame would move to before the first event, then the frame must shrink.

Top case: If the current iterator is the top (of the frame) iterator, then the top iterator needs to be incremented. The new top event has the same index as the now-gone top event. The index of the bottom event is decremented, since an event before it is now gone. The bottom iterator moves to the next event, which is now at the bottom of the frame. The current event is treated like the top event.

Inside case: If the current iterator is in the middle of the frame, the top iterator and index remain unchanged. The current iterator is incremented, but its index is now the same as the old bottom index. Same for the bottom iterator.

Bottom case: If the current iterator (and bottom iterator) point to the last event in the frame, then both of them need to be decremented. The frame needs to be moved up by one event, so that the current event remains at the bottom (it's just simpler to manage that way).

If there is no event after the bottom of the frame, the iterators that now point to end() must backtrack one event. If the container becomes empty, then everything is invalidated.

Returns

Returns true if the delete was possible. If the container was empty or became empty, then false is returned.

12.12.1.6 `bool seq64::eventslots::modify_current_event (const std::string & evtimestamp, const std::string & evname, const std::string & evdata0, const std::string & evdata1) [private]`

If the timestamp has changed, however, we can't just modify the event in place. Instead, we finish modifying the event, but tell the caller to delete and reinsert the new event (in its proper new location based on timestamp).

This function always copies the original event, modifies the copy, deletes the original event, and inserts the "new" event into the editable-event container.

Parameters

<i>evtimestamp</i>	Provides the new event time-stamp as edited by the user.
<i>evname</i>	Provides the event name as edited by the user.
<i>evdata0</i>	Provides the time-stamp as edited by the user.
<i>evtimestamp</i>	Provides the time-stamp as edited by the user.

Returns

Returns true simply if the event-count is greater than 0.

12.12.1.7 `bool seq64::eventslots::save_events () [private]`

Also sets the dirty flag for the sequence, via the [sequence::add_event\(\)](#) function, but this doesn't seem to set the perform dirty flag. So now we pass the modification buck to the parent, who passes it to the perform object.

We added a `copy_events()` function in the sequence class to replace `add_event()` for the purpose of reconstructing the events container for the sequence. It is locked by a mutex, and so will not draw until all is done, preventing a nasty segfault (all segfaults are nasty).

We create a new plain event container here, and then passing it to the new locked/threadsafe `sequence::copy_events()` function that clears the sequence container and copies the events from the parameter container.

Note that this code will operate event if all events were deleted.

Returns

Returns true if the operations succeeded.

12.12.1.8 `void seq64::eventslots::select_event (int event_index = SEQ64_NULL_EVENT_INDEX, bool full_redraw = true) [private]`

The event index is provided by converting the y-coordinate of the mouse pointer into a slot number, and then an event index (actually the slot-distance from the `m_top_iterator`. Confusing, yes no?

Note that, if the event index is negative, then we just queue up a draw operation, which should paint an empty frame – the event container is empty.

Parameters

<i>event_index</i>	Provides the numeric index of the event in the event frame, or <code>SEQ64_NULL_EVENT</code> if there is no event to draw.
<i>full_redraw</i>	Defaulting to true, this parameter can be set to false in some case to reduce the flickering of the frame under fast movement.

12.12.1.9 `void seq64::eventslots::set_text (const std::string & evcategory, const std::string & evtimestamp, const std::string & evname, const std::string & evdata0, const std::string & evdata1) [private]`

Parameters

<i>evtimestamp</i>	The event time-stamp to be set in the parent.
<i>evname</i>	The event name to be set in the parent.
<i>evdata0</i>	The first event data byte to be set in the parent.
<i>evdata1</i>	The second event data byte to be set in the parent.

12.12.1.10 `int seq64::eventslots::convert_y (int y) [private]`

Parameters

<i>y</i>	The y coordinate of the position of the mouse click in the eventslot window/pixmap.
----------	---

Returns

Returns the index of the event position in the user-interface, which should range from 0 to `m_line_count`.

12.12.1.11 `void seq64::eventslots::draw_event (editable_events::iterator ei, int index)` [private]

The slot contains the event details in (so far) one line of text in the box:

| timestamp | event kind | channel | data 0 name + value | data 1 name + value

Currently, this view shows only events that get copied to the sequence's event list. This rules out the following items from the view:

- MThd (song header)
- MTrk and Meta TrkEnd (track marker, a sequence has only one track)
- SeqNr (sequence number)
- SeqSpec (but there are three that might appear, see below)
- Meta TrkName

The events that are shown in this view are:

- One-data-value events:
 - Program Change
 - Channel Pressure
- Two-data-value events:
 - Note Off
 - Note On
 - Aftertouch
 - Control Change
 - Pitch Wheel
- Other:
 - SysEx events, with partial show of data bytes
 - SeqSpec events (TBD):
 - Key
 - Scale
 - Background sequence

The index of the event is shown in the editor portion of the eventedit dialog.

12.12.1.12 `void seq64::eventslots::draw_events ()` [private]

It first clears the whole bitmap to white, so that no artifacts from the previous state of the frame are left behind.

Need to figure out how to calculate the number of displayable events.

```
m_line_maximum = ???
```

12.12.1.13 `void seq64::eventslots::change_vert ()` [private]

Note that `m_vadjust` is the `Gtk::Adjustment` object that the eventedit parent passes to the `gui_drawingarea_gtk2` constructor.

The top-event and bottom-event indices (and their corresponding editable-event iterators) delimit the part of the event container that is displayed in the eventslots user-interface. The top-event index starts at 0, and the bottom-event is larger (initially, by 42 slots).

When the scroll-bar thumb moves up or down, we need to change both event indices and both event iterators by the corresponding amount. Luckily, the `std::multimap` iterator is bidirectional.

Note that we may need to reduce the movement of events to a value less than a page; it can be limited backwards by the value of the top index, and forward by the value of the bottom index.

12.12.1.14 `void seq64::eventslots::page_movement (int new_value)` [private]

The adjustment is done via movement from the current position.

Do we even need a way to detect excess movement? The scrollbar, if properly set up, should never move the frame too high or too low. Verified by testing.

Parameters

<i>new_value</i>	Provides the new value of the scrollbar position.
------------------	---

12.12.1.15 void seq64::eventslots::page_topper (editable_events::iterator *newcurrent*) [private]

The adjustment is done "from scratch". We've found page movement to be an insoluable problem in some editing circumstances. So now we move to the inserted event, and make it the top event.

However, always moving an inserted event to the top is a bit annoying. So now we backtrack so that the inserted event is at the bottom.

Parameters

<i>newcurrent</i>	Provides the iterator to the event to be shown at the bottom of the frame.
-------------------	--

12.12.1.16 int seq64::eventslots::decrement_top () [private]

Returns

Returns 0, or SEQ64_NULL_EVENT_INDEX if the iterator could not be decremented.

12.12.1.17 int seq64::eventslots::increment_top () [private]

Also handles the top-event index, so that the GUI can display the proper event numbers.

Returns

Returns the top index, or SEQ64_NULL_EVENT_INDEX if the iterator could not be incremented, or would increment to the end of the container.

12.12.1.18 int seq64::eventslots::decrement_current () [private]

Returns

Returns the decremented index, or SEQ64_NULL_EVENT_INDEX if the iterator could not be decremented. Remember that the index ranges only from 0 to m_line_count-1, and that is enforced here.

12.12.1.19 int seq64::eventslots::increment_current () [private]

Returns

Returns the incremented index, or SEQ64_NULL_EVENT_INDEX if the iterator could not be incremented. Remember that the index ranges only from 0 to m_line_count-1, and that is enforced here.

12.12.1.20 `int seq64::eventslots::decrement_bottom () [private]`

Returns

Returns 0, or SEQ64_NULL_EVENT_INDEX if the iterator could not be decremented.

12.12.1.21 `int seq64::eventslots::increment_bottom () [private]`

There is an issue in paging down using the scrollbar where, at the bottom of the scrolling, the bottom iterator ends up bad. Not yet sure how this happens, so for now we backtrack one event if this happens.

Returns

Returns the incremented index, or SEQ64_NULL_EVENT_INDEX if the iterator could not be incremented.

12.12.1.22 `void seq64::eventslots::on_realize () [private]`

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

12.12.1.23 `bool seq64::eventslots::on_expose_event (GdkEventExpose * ev) [private]`

It draws all of the sequences.

12.12.1.24 `bool seq64::eventslots::on_focus_in_event (GdkEventFocus * ev) [private]`

See the same function in the perfrroll module.

12.12.1.25 `void seq64::eventslots::on_size_allocate (Gtk::Allocation & a) [private]`

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

12.12.1.26 `void seq64::eventslots::on_move_up () [private]`

We must scroll up if the event is now before the frame, and should be made the new top event of the frame. Note that this function isn't really an event-response callback. It is called by [eventedit::on_key_press_event\(\)](#).

12.12.1.27 `void seq64::eventslots::on_move_down () [private]`

We must scroll down if the event is now after the frame. Note that this function isn't really an event-response callback. It is called by [eventedit::on_key_press_event\(\)](#).

12.12.2 Field Documentation

12.12.2.1 `int seq64::eventslots::m_slots_chars` `[private]`

Pretty much hardwired to 24 at present.

12.12.2.2 `int seq64::eventslots::m_char_w` `[private]`

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

12.12.2.3 `int seq64::eventslots::m_setbox_w` `[private]`

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

12.12.2.4 `int seq64::eventslots::m_slots_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.

12.12.2.5 `int seq64::eventslots::m_xy_offset` `[private]`

Currently hardwired.

12.12.2.6 `int seq64::eventslots::m_top_index` `[private]`

It is used in numbering the events that are shown in the event-slot frame. Do not confuse it with `m_current_index`, which is relative to the frame, not the container-beginning.

12.12.2.7 `int seq64::eventslots::m_current_index` `[private]`

This event will also be pointed to by the `m_current_event` iterator. Do not confuse it with `m_top_index`, which is relative to the container-beginning, not the frame.

12.13 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

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_y.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_y_on_b_pixmap](#)
The pixmap in the file src/pixmaps/font_yb.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_b_on_c_pixmap](#)
The pixmap in the file src/pixmaps/cyan_wenfont_y.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Pixmap > [m_c_on_b_pixmap](#)
The pixmap in the file src/pixmaps/cyan_wenfont_yb.xpm is loaded into this object.
- Glib::RefPtr< Gdk::Bitmap > [m_clip_mask](#)
This object is instantiated as a default object.

12.13.1 Member Enumeration Documentation

12.13.1.1 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.

BLACK_ON_CYAN A new color, for drawing black text on a cyan background.

CYAN_ON_BLACK A new color, for drawing cyan text on a black background.

12.13.2 Member Function Documentation

12.13.2.1 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.

12.13.2.2 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.

12.13.3 Field Documentation

12.13.3.1 `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.

12.13.3.2 `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.

12.13.3.3 `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).

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

It contains a black font on a white background. The new-style font, if selected, is in the `resources/pixmaps/wenfont←_b.xmp` pixmap.

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

It contains a black font on a white background. The new-style font, if selected, is in the `resources/pixmaps/wenfont←_w.xmp` pixmap.

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

It contains a black font on a yellow background. The new-style font, if selected, is in the `resources/pixmaps/wenfont←_y.xmp` pixmap.

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

It contains a yellow font on a black background. The new-style font, if selected, is `resources/pixmaps/wenfont←_yb.xmp` pixmap.

12.13.3.8 `Glib::RefPtr<Gdk::Pixmap> seq64::font::m_b_on_c_pixmap` `[private]`

It contains a black font on a cyan background. It is available only for the new font-style.

12.13.3.9 `Glib::RefPtr<Gdk::Pixmap> seq64::font::m_c_on_b_pixmap` `[private]`

It contains a cyan font on a black background. It is available only for the new font-style.

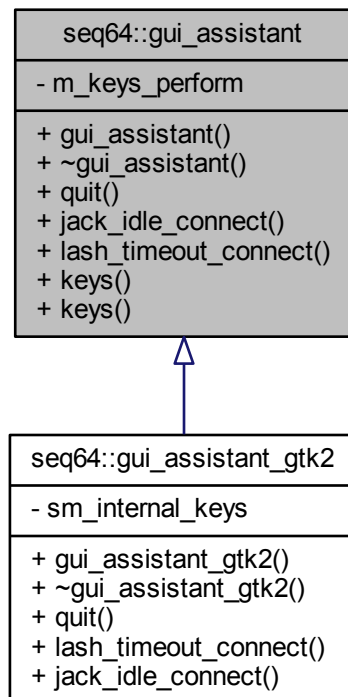
12.13.3.10 Glib::RefPtr<Gdk::Bitmap> seq64::font::m_clip_mask [private]

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

12.14 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.

12.14.1 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).

12.14.2 Constructor & Destructor Documentation

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

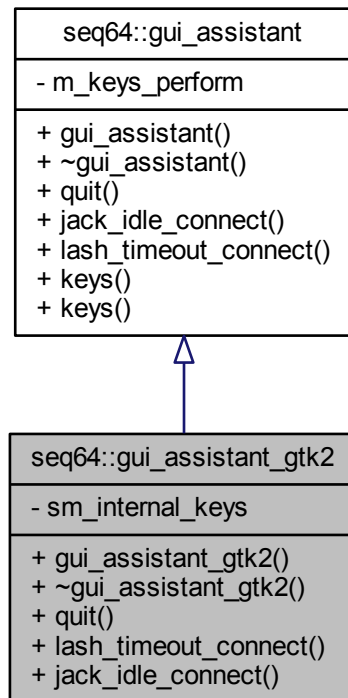
Parameters

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

12.15 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 [lash_timeout_connect \(lash *lashobject\)](#)
Connects the LASH timeout-event callback to the Glib timeout object.
- virtual void [jack_idle_connect \(jack_assistant &jack\)](#)
Connects the JACK session-event callback to the Glib idle object.

Static Private Attributes

- static [keys_perform_gtk2 sm_internal_keys](#)
Provides a pre-made [keys_perform](#) object.

12.15.1 Member Function Documentation

12.15.1.1 void seq64::gui_assistant_gtk2::lash_timeout_connect (lash * *lashobject*) [virtual]

The time-out value is set to 250 ms.

Implements [seq64::gui_assistant](#).

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 [scroll_adjust](#) (Gtk::Adjustment &adjust, double step)

This function provides optimization for the `on_scroll_event()` functions, and should provide support for having the `seqedit/seqroll/seqtime/seqdata` panes follow the scrollbar, in a future upgrade.

 - 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

12.16.1 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.

12.16.2 Member Function Documentation

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

One less need to access `m_window` directly.

12.16.2.2 void `seq64::gui_drawingarea_gtk2::set_line` (`Gdk::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.

12.16.2.3 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.

12.16.2.4 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.

12.16.2.5 `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.

12.16.2.6 `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.

12.16.2.7 `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.

12.16.2.8 `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>pixmap</i>	Provides the Gdk::Drawable pointer needed to draw the line.
<i>c</i>	The foreground color in which to draw the line.

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.

12.16.2.9 `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.

12.16.2.10 `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.

12.16.2.11 `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.

12.16.2.12 `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.

Parameters

<i>s</i>	The string to be drawn.
<i>color</i>	The color with which to draw the string.

12.16.2.13 `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.

12.16.2.14 `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.

12.16.2.15 `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.

12.16.2.16 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.

12.16.2.17 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>pixmap</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.

12.16.2.18 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.

12.16.2.19 `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.

12.16.2.20 `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.

12.16.2.21 `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.

12.16.2.22 `void seq64::gui_drawingarea_gtk2::scroll_adjust (Gtk::Adjustment & hadjust, double step)`
`[protected]`

This function is currently duplicated in the [gui_drawingarea_gtk2](#) and [gui_window_gtk2](#) modules.

Parameters

<i>hadjust</i>	Provides a reference to the adjustment object to be adjusted.
----------------	---

Parameters

<i>step</i>	Provides the step value to use for adjusting the horizontal scrollbar. If negative, the adjustment is leftward. If positive, the adjustment is rightward. It can be the value of <code>m_hadjust->get_step_increment()</code> , or provided especially to keep up with the progress bar.
-------------	---

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

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

12.16.3 Field Documentation

12.16.3.1 `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.

12.16.3.2 `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.

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

Our wrappers still leave this member exposed <giggle>.

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

Our wrappers still leave this member exposed.

12.16.3.5 `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.

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

Could make this constant, but some windows are resizable.

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

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

- 'Getter' function for member m_red*
- const [Color](#) & [orange](#) () const
- 'Getter' function for member m_orange*
- const [Color](#) & [yellow](#) () const
- 'Getter' function for member m_yellow*
- const [Color](#) & [green](#) () const
- 'Getter' function for member m_green*
- const [Color](#) & [blue](#) () const
- 'Getter' function for member m_blue*
- const [Color](#) & [dark_cyan](#) () const
- 'Getter' function for member m_dk_cyan*
- const [Color](#) & [bg_color](#) () const
- 'Getter' function for member m_bg_color*
- void [bg_color](#) (const [Color](#) &c)
- 'Setter' function for member m_bg_color*
- const [Color](#) & [fg_color](#) () const
- 'Getter' function for member m_fg_color*
- void [fg_color](#) (const [Color](#) &c)
- 'Setter' function for member m_fg_color*

Protected Types

- typedef Gdk::Color [Color](#)
Provides a type for the color object.

12.17.1 Detailed Description

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

12.17.2 Constructor & Destructor Documentation

12.17.2.1 seq64::gui_palette_gtk2::gui_palette_gtk2 ()

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

12.17.3 Member Function Documentation

12.17.3.1 const [Color](#)& seq64::gui_palette_gtk2::line_color () const [inline]

Might eventually be selectable from the "user" configuration file

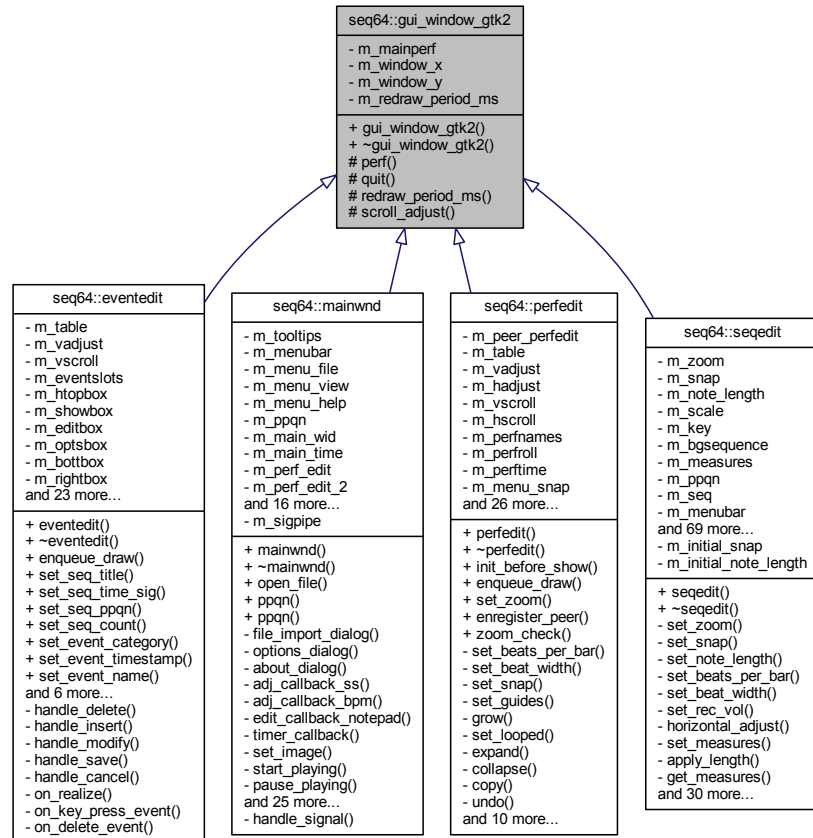
12.17.3.2 const [Color](#)& seq64::gui_palette_gtk2::progress_color () const [inline]

Might eventually be selectable from the "user" configuration file

12.18 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
- virtual void [quit](#) ()
Provides "quit" functionality that WE HAVE OVERLOOKED!!!
- int [redraw_period_ms](#) () const
'Getter' function for member m_redraw_period_ms
- void [scroll_adjust](#) (Gtk::Adjustment &adjust, double step)
This function provides optimization for the on_scroll_event() functions, and should provide support for having the seqedit/seqroll/seqtime/seqdata panes follow the scrollbar, in a future upgrade.

Private Attributes

- [perform](#) & [m_mainperf](#)
The master object, sort of a sequence buss.
- `int` [m_window_x](#)
Window sizes.
- `int` [m_redraw_period_ms](#)
Provides the timer period for the eventedit timer, used to determine the rate of redrawing.

12.18.1 Constructor & Destructor Documentation

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

Note

We've collected the redraw timeouts into a base-class member. Most were valued at `c_redraw_ms` (40 ms), but `mainwnd` used 25 ms, so beware. We will eventually make this a user-interface parameter.

Parameters

<i>p</i>	Refers to the main performance object.
<i>window</i> ↔ <i>_x</i>	The width of the window.
<i>window</i> ↔ <i>_y</i>	The height of the window.

12.18.2 Member Function Documentation

12.18.2.1 `void seq64::gui_window_gtk2::scroll_adjust (Gtk::Adjustment & hadjst, double step)` `[protected]`

This function is currently duplicated in the [gui_drawingarea_gtk2](#) and [gui_window_gtk2](#) modules.

Parameters

<i>hadjst</i>	Provides a reference to the adjustment object to be adjusted.
<i>step</i>	Provides the step value to use for adjusting the horizontal scrollbar. If negative, the adjustment is leftward. If positive, the adjustment is rightward. It can be the value of <code>m_hadjst->get_step_increment()</code> , or provided especially to keep up with the progress bar.

12.18.3 Field Documentation

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

Could make this constant, but some windows are resizable.

12.18.3.2 int seq64::gui_window_gtk2::m_redraw_period_ms [private]

This is currently hardwired to 40 ms in Linux, and 20 ms in Windows. Note that mainwnd used 25 ms.

12.19 seq64::jack_assistant Class Reference

This class provides the performance mode JACK support.

Public Member Functions

- [jack_assistant](#) ([perform](#) & [parent](#), int bpmminute=SEQ64_DEFAULT_BPM, int ppqn=SEQ64_USE_DEFAULT_PPQN, int bpm=SEQ64_DEFAULT_BEATS_PER_MEASURE, int beatwidth=SEQ64_DEFAULT_BEAT_WIDTH)
 - This constructor initializes a number of member variables, some of them public!*
- [~jack_assistant](#) ()
 - The destructor doesn't need to do anything yet.*
- [perform](#) & [parent](#) ()
 - 'Getter' function for member m_jack_parent Needed for external callbacks.*
- bool [is_running](#) () const
 - 'Getter' function for member m_jack_running*
- bool [is_master](#) () const
 - 'Getter' function for member m_jack_master*
- int [get_ppqn](#) () const
 - 'Getter' function for member m_ppqn*
- int [get_beat_width](#) () const
 - 'Getter' function for member m_beat_width*
- void [set_beat_width](#) (int bw)
 - 'Setter' function for member m_beat_width*
- int [get_beats_per_measure](#) () const
 - 'Getter' function for member m_beats_per_measure*
- void [set_beats_per_measure](#) (int bpm)
 - 'Setter' function for member m_beats_per_measure*
- int [get_beats_per_minute](#) () const
 - 'Getter' function for member m_beats_per_minute*
- void [set_beats_per_minute](#) (int bpmminute)
 - 'Setter' function for member m_beats_per_minute For the future, changing the BPM (beats/minute) internally.*
- bool [init](#) ()
 - Initializes JACK support.*
- bool [deinit](#) ()
 - Tears down the JACK infrastructure.*
- bool [session_event](#) ()
 - Writes the MIDI file named "<jack session dir>-file.mid" using a midifile object, quits if told to by JACK, and can free the JACK session event.*
- void [start](#) ()
 - If JACK is supported, starts the JACK transport.*
- void [stop](#) ()
 - If JACK is supported, stops the JACK transport.*
- void [position](#) (bool to_left_tick, bool relocate=false)

If JACK is supported and running, sets the position of the transport to the new frame number, frame 0.

- bool [output](#) ([jack_scratchpad](#) &pad)
Performance output function for JACK, called by the perform function of the same name.
- void [set_ppqn](#) (int ppqn)
'Setter' function for member m_ppqn For the future, changing the PPQN internally.
- double [get_jack_tick](#) () const
'Getter' function for member m_jack_tick
- const jack_position_t & [get_jack_pos](#) () const
'Getter' function for member m_jack_pos

Private Member Functions

- void [set_jack_running](#) (bool flag)
'Setter' function for member m_jack_running
- jack_client_t * [client](#) () const
'Getter' function for member m_jack_client
- bool [info_message](#) (const std::string &msg)
Common-code for console messages.
- bool [error_message](#) (const std::string &msg)
Common-code for error messages.
- jack_client_t * [client_open](#) (const std::string &clientname)
A more full-featured initialization for a JACK client, which is meant to be called by the [init\(\)](#) function.
- void [show_statuses](#) (unsigned bits)
Loops through the full set of JACK bits, showing the information for any bits that are set in the given parameter.
- void [show_position](#) (const jack_position_t &pos) const
Shows a one-line summary of a JACK position structure.
- int [sync](#) (jack_transport_state_t state=(jack_transport_state_t)(-1))
A helper function for syncing up with JACK parameters.
- void [set_position](#) (midipulse currenttick)
Provides the code that was effectively commented out in the [perform::position_jack\(\)](#) function.

Static Private Attributes

- static jack_status_pair_t [sm_status_pairs](#) []
Provides a list of JACK status bits, and a brief string to explain the status bit.

Friends

- void [jack_shutdown_callback](#) (void *arg)
This callback is to shutdown JACK by clearing the jack_assistant::m_jack_running flag.
- 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_timebase_callback](#) (jack_transport_state_t state, jack_nframes_t nframes, jack_position_t *pos, int new_pos, void *arg)
The JACK timebase function defined here sets the JACK position structure.
- void [jack_session_callback](#) (jack_session_event_t *ev, void *arg)
Set the m_jsession_ev (event) value of the perform object.

12.19.1 Constructor & Destructor Documentation

12.19.1.1 `seq64::jack_assistant::jack_assistant (perform & parent, int bpmminute = SEQ64_DEFAULT_BPM, int ppqn = SEQ64_USE_DEFAULT_PPQN, int bpm = SEQ64_DEFAULT_BEATS_PER_MEASURE, int beatwidth = SEQ64_DEFAULT_BEAT_WIDTH)`

Note that the perform object currently calls `jack_assistant::init()`, but that call could be made here instead.

Parameters

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

12.19.1.2 `seq64::jack_assistant::~~jack_assistant ()`

The perform object currently calls `jack_assistant::deinit()`, but that call could be made here instead.

12.19.2 Member Function Documentation

12.19.2.1 `void seq64::jack_assistant::set_beats_per_minute (int bpmminute) [inline]`

We should consider adding validation. However, `perform::set_beats_per_minute()` does validate already.

12.19.2.2 `bool seq64::jack_assistant::init ()`

Then we become a new client of the JACK server.

A sync callback is needed for polling of slow-sync clients. But seq24/sequencer64 are not slow-sync clients. We don't really need to be a slow-sync client, as far as we can tell. We can't get JACK working exactly the way it does in seq24 without the callback in place. Plus, it does things important to the setup of JACK. So now this setup is permanent.

Jack transport settings:

```
There are three settings: On, Master, and Master Conditional.
Currently, they can all be selected in the user-interface's File /
Options / JACK/LASH page. We really want only the proper combinations
to be set, for clarity (the user-interface now takes care of this. We
need to initialize if any of them are set, and the
rc_settings::with_jack() function tells us that.
```

`jack_set_process_callback()` patch:

```
Implemented first patch from freddix/seq24 GitHub project, to fix JACK
transport. One line of code. Well, we added some error-checking. :-)
Found some old notes on the Web the this patch really only works (to
prevent seq24 freeze) if seq24 is set as JACK Master, or if another
client application, such as Qtractor, is running as JACK Master (and
then seq24 will apparently follow it).
```

Returns

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

12.19.2.3 `bool seq64::jack_assistant::deinit ()`

Returns the value of `m_jack_running`, which should be false.

12.19.2.4 `bool seq64::jack_assistant::session_event ()`

ca 2015-07-24 Just a note: The OMA (OpenMandrivaAssociation) patch was already applied to seq24 v.0.9.2. It put quotes around the `-file` argument. However, the `-file` option doesn't work, so let's change that line.

```
sequencer64 --file \"${SESSION_DIR}file.mid\" --jack_session_uuid
```

Why are we using a `Glib::ustring` here? Convenience. But with C++11, we could use a `lexical_cast<>`. No more `ustring`, baby! It doesn't really matter; this function can call `Gtk::Main::quit()`, via the parent's `gui().quit()` function.

Returns

Always returns false.

12.19.2.5 `void seq64::jack_assistant::start ()`

This function assumes that `m_jack_client` is not null, if `m_jack_running` is true.

Found this note in the Hydrogen code:

```
When jack_transport_start() is called, it takes effect from the next
processing cycle. The location info from the timebase_master, if
there is one, will not be available until the _next_ next cycle. The
code must therefore wait one cycle before syncing up with
timebase_master.
```

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

This function assumes that `m_jack_client` is not null, if `m_jack_running` is true.

12.19.2.7 `void seq64::jack_assistant::position (bool to_left_tick, bool relocate = false)`

This new position takes effect in two process cycles. If there are slow-sync clients and the transport is already rolling, it will enter the `JackTransportStarting` state and begin invoking their `sync_callbacks` until ready. This function is realtime-safe.

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

This `position()` function is called via `perform::position_jack()` in the `mainwnd`, `perfed`, `perfr`, and `seqroll` graphical user-interface support objects.

The code that was disabled sets the current tick to 0 or, if state was true, to the leftmost tick (which is probably the position of the L marker). The current tick is then converted to a frame number, and then we locate the transport to that position. We're going to enable this code, but make it dependent on a new boolean parameter that defaults to false, in anticipation of trying it out later.

These repositions reset the progress bars. We don't always want that, it should be an option. TODO.

`jack_transport_reposition()`:

Requests a new transport position. The new position takes effect in two process cycles. If there are slow-sync clients and the transport is already rolling, it will enter the JackTransportStarting state and begin invoking their sync_callbacks until ready. This function is realtime-safe.

It's pos parameter provides the requested new transport position. Fill pos->valid to specify which fields should be taken into account. If you mark a set of fields as valid, you are expected to fill them all. Note that "frame" is always assumed, and generally needs to be set:

<http://comments.gmane.org/gmane.comp.audio.jackit/18705>

Returns 0 if a valid request, EINVAL if the position structure is rejected.

Warning

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

Parameters

<i>to_left_tick</i>	If true, the current tick is set to the leftmost tick, instead of the 0th tick. Now used, but only if relocate is true. One question is, do we want to perform this function if rc().with_jack_transport() is true? Seems like we should be able to do it only if m_jack_master is true.
<i>relocate</i>	If true (it defaults to false), then we allow the relocation of the JACK transport to the current_tick or the left tick, rather than to frame 0. EXPERIMENTAL, enables dead code from seq24. Seems to work if set to true when we are the JACK Master. Enabling this code makes "klick -j -P" work, after a fashion. It clicks, but at a way too rapid rate.

12.19.2.8 bool seq64::jack_assistant::output (jack_scratchpad & pad)

This code comes from [perform::output_func\(\)](#) from seq24.

Note

Follow up on this note found "out there": "Maybe I'm wrong but if I understood correctly, recent jack1 transport no longer goes into Jack_Transport_Starting state before going to Jack_Transport_Rolling (this was deliberately dropped), but seq24 currently needs this to start off with jack transport." On the other hand, some people have no issues. This may have been due to the lack of m_jack_pos initialization.

Parameters

<i>pad</i>	Provide a JACK scratchpad for sharing certain items between the perform object and the jack_assistant object.
------------	---

Returns

Returns true if JACK is running.

12.19.2.9 void seq64::jack_assistant::set_ppqn (int ppqn) [inline]

We should consider adding validation. But it is used by perform.

12.19.2.10 `bool 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.
------------	---

Returns

Returns true.

12.19.2.11 `bool 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.
------------	---

Returns

Returns false for convenience/brevity in setting function return values.

12.19.2.12 `jack_client_t * seq64::jack_assistant::client_open (const std::string & clientname)` [private]

Status bits for `jack_status_t` return pointer:

`JackNameNotUnique` means that the client name was not unique. With `JackUseExactName`, this is fatal. Otherwise, the name was modified by appending a dash and a two-digit number in the range "-01" to "-99". The `jack_get_client_name()` function returns the exact string used. If the specified `client_name` plus these extra characters would be too long, the open fails instead.

`JackServerStarted` means that the JACK server was started as a result of this operation. Otherwise, it was running already. In either case the caller is now connected to jackd, so there is no race condition. When the server shuts down, the client will find out.

Returns

Returns true if JACK has opened the client connection successfully.

12.19.2.13 `void seq64::jack_assistant::show_statuses (unsigned bits) [private]`

For reference, here are the enumeration values from `/usr/include/jack/types.h`:

```

JackFailure           = 0x01
JackInvalidOption     = 0x02
JackNameNotUnique     = 0x04
JackServerStarted     = 0x08
JackServerFailed      = 0x10
JackServerError       = 0x20
JackNoSuchClient      = 0x40
JackLoadFailure       = 0x80
JackInitFailure       = 0x100
JackShmFailure        = 0x200
JackVersionError      = 0x400
JackBackendError      = 0x800
JackClientZombie      = 0x1000

```

12.19.2.14 `void seq64::jack_assistant::show_position (const jack_position_t & pos) const [private]`

This function is meant for experimenting and learning.

The fields of this structure are as follows. Only the fields we care about are shown.

```

jack_nframes_t      frame_rate:    current frame rate (per second)
jack_nframes_t      frame:         frame number, always present
jack_position_bits_t valid:        which other fields are valid

```

JackPositionBBT:

```

int32_t             bar:           current bar
int32_t             beat:          current beat-within-bar
int32_t             tick:          current tick-within-beat
double              bar_start_tick
float               beats_per_bar: time signature "numerator"
float               beat_type:     time signature "denominator"
double              ticks_per_beat
double              beats_per_minute

```

JackBBTFrameOffset:

```

jack_nframes_t      bbt_offset;    frame offset for the BBT fields

```

Only the most "important" and time-varying fields are shown. The format output is brief and inscrutable unless you read this format example:

```

nnnnn frame B:B:T N/D TPB BPM BBT
^   ^   ^   ^ ^ ^   ^   ^
|   |   |   | | |   |   |
|   |   |   | | |   |   | ----- bbt_offset (frame), even if invalid
|   |   |   | | |   |   | ----- beats_per_minute
|   |   |   | | |   |   | ----- ticks_per_beat (PPQN * 10?)
|   |   |   | | |   |   | ----- beat_type (denominator)
|   |   |   | | |   |   | ----- beats_per_bar (numerator)
|   |   |   | | |   |   | ----- bar : beat : tick
|   |   |   | | |   |   | ----- frame (number)
|   |   |   | | |   |   | ----- the "valid" bits
-----

```

The "valid" field is shown as bits in the same bit order as shown here, but represented as a five-character string, "nnnnn", n = 0 or 1:

```

JackVideoFrameOffset = 0x100
JackAudioVideoRatio  = 0x080
JackBBTFrameOffset   = 0x040
JackPositionTimecode = 0x020
JackPositionBBT      = 0x010

```

We care most about nnnnn = "00101" in our experiments (the most common output will be "00001"). And we don't worry about non-integer measurements... we truncate them to integers. Change the output format if you want to play with non-Western timings.

Parameters

<i>pos</i>	The JACK position structure to dump.
------------	--------------------------------------

```
12.19.2.15 int seq64::jack_assistant::sync ( jack_transport_state_t state = (jack_transport_state_t) (-1) )
           [private]
```

Sequencer64 is not a slow-sync client, so that callback is not really needed, but we probably need this sub-function here to start out with the right values for interacting with JACK.

Note the call to `jack_transport_query()`. This call is *not* in seq24, but seems to be needed in sequencer64 because we put `m_jack_pos` in the initializer list, which sets all its fields to 0. Seq24 accesses `m_jack_pos` before it ever gets set, but its fields have values. These values are bogus, but are consistent from run to run on my computer, and allow seq24 to follow another JACK Master, on some computers. It explains why people had different experiences with JACK sync.

If we explicitly call `jack_transport_query()` here, without changing the *state* parameter, then sequencer64 also can follow another JACK Master. (CURRENTLY BUGGY!)

Note that we should consider massaging the following `jack_position_t` members to set them to 0 (or 0.0) if less than 1.0 or 0.5:

```
- bar_start_tick
- ticks_per_beat
- beats_per_minute
- frame_time
- next_time
- audio_frames_per_video_frame
```

Also, why does `bbt_offset` start at 2128362496?

```
12.19.2.16 void seq64::jack_assistant::set_position ( midipulse currenttick ) [private]
```

We might be able to use it in other functions.

Computing the BBT information from the frame number is relatively simple here, but would become complex if we supported tempo or time signature changes at specific locations in the transport timeline.

```
ticks * 10 = jack ticks;
jack ticks / ticks per beat = num beats;
num beats / beats per minute = num minutes
num minutes * 60 = num seconds
num seconds * frame_rate = frame
```

12.19.3 Friends And Related Function Documentation

```
12.19.3.1 void jack_shutdown_callback ( void * arg ) [friend]
```

Parameters

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

12.19.3.2 `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.

The transport state will be:

- JackTransportStopped when a new position is requested.
- JackTransportStarting when the transport is waiting to start.
- JackTransportRolling when the timeout has expired, and the position is now a moving target.

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.

Returns

Returns 1 if the function works, and 0 if something was wrong.

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

The original version of the function worked properly with Hydrogen, but not with Klick. The new code seems to work with both. More testing and clarification is needed. This new code was "discovered" in the source-code for the "SooperLooper" project:

<http://esseg.net/sooperlooper/>

The first difference with the new code is that it handles the case where the JACK position is moved (`new_pos == true`). If this is true, and the JackPositionBBT bit is off in `pos->valid`, then the new BBT value is set.

The seconds set of differences are in the "else" clause. In the new code, it is very simple: calculate the new tick value, back it off by the number of ticks in a beat, and perhaps go to the first beat of the next bar.

In the old code (complex!), the simple BBT adjustment is always made. This changes (perhaps) the `beats_per_bar`, `beat_type`, etc. We need to make these settings use the actual global values for beats set for Sequencer64. Then, if transitioning from JackTransportStarting to JackTransportRolling (instead of checking `new_pos!`), the BBT values (bar, beat, and tick) are finally adjusted. Here are the steps, with old and new steps noted:

```
-# Calculate the "delta" ticks based on the current frame, the
  ticks_per_beat, the beats_per_minute, and the frame_rate. The old
  code saves this in a local, the new code assigns it to pos->tick.
-# Old code: save this delta as a positive value.
-# Figure out the settings and modify bar, beat, tick, and
  bar_start_tick. The old and new code seem to have the same intent,
  but it seems like the new code is faster and also correct.
  - Old code: Calculations are made by division and mod
    operations.
  - New code: Calculations are made by increments and decrements
    in a while loop.
```

Parameters

<i>state</i>	Indicates the current state of JACK transport.
<i>nframes</i>	The number of JACK frames in the current time period.
<i>pos</i>	Provides the position structure to be filled in, the address of the position structure for the next cycle; pos->frame will be its frame number. If new_pos is FALSE, this structure contains extended position information from the current cycle. If TRUE, it contains whatever was set by the requester. The timebase_callback's task is to update the extended information here.
<i>new_pos</i>	TRUE (non-zero) for a newly requested pos, or for the first cycle after the timebase_callback is defined. This is usually 0 in Sequencer64 at present, and 1 if one, say, presses "rewind" in qjackctl.
<i>arg</i>	Provides the jack_assistant pointer, currently unchecked for nullity.

12.19.3.4 void jack_session_callback (jack_session_event_t * ev, void * arg) [friend]

Glib is then used to connect in perform::jack_session_event(). However, the perform object's GUI-support interface is used instead of the following, so that the libseq64 library can be independent of a specific GUI framework:

```
Glib::signal_idle().
    connect (sigc::mem_fun (*jack, &jack_assistant::session_event));
```

Parameters

<i>ev</i>	The JACK event to be set.
<i>arg</i>	The pointer to the jack_assistant object. Currently not checked for nullity.

12.19.4 Field Documentation

12.19.4.1 jack_status_pair_t seq64::jack_assistant::sm_status_pairs [static],[private]

Terminated by a 0 value and an empty string.

12.20 seq64::jack_scratchpad Class Reference

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

12.20.1 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\(\)](#). This scratchpad is useful even if JACK support is not enabled.

12.21 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

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???

12.21.1 Member Enumeration Documentation

12.21.1.1 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.

12.21.2 Constructor & Destructor Documentation

12.21.2.1 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.

12.21.3 Member Function Documentation

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

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

12.21.3.2 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.

12.21.4 Field Documentation

12.21.4.1 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.

12.22 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.

12.22.1 Detailed Description

It provides a way a mapping keystrokes to sequencer actions and song settings.

12.22.2 Constructor & Destructor Documentation

12.22.2.1 seq64::keys_perform::~~keys_perform ()

Finally, any active patterns/sequences are deleted.

12.22.3 Member Function Documentation

12.22.3.1 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.
------------	--

12.22.3.2 `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.
------------	--

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

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

12.22.3.4 `bool seq64::keys_perform::show_ui_sequence_number () const [inline]`

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

12.22.3.5 `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](#).

12.22.3.6 `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](#).

12.22.3.7 `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](#).

12.22.3.8 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.

12.22.3.9 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.

12.22.4 Field Documentation

12.22.4.1 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).

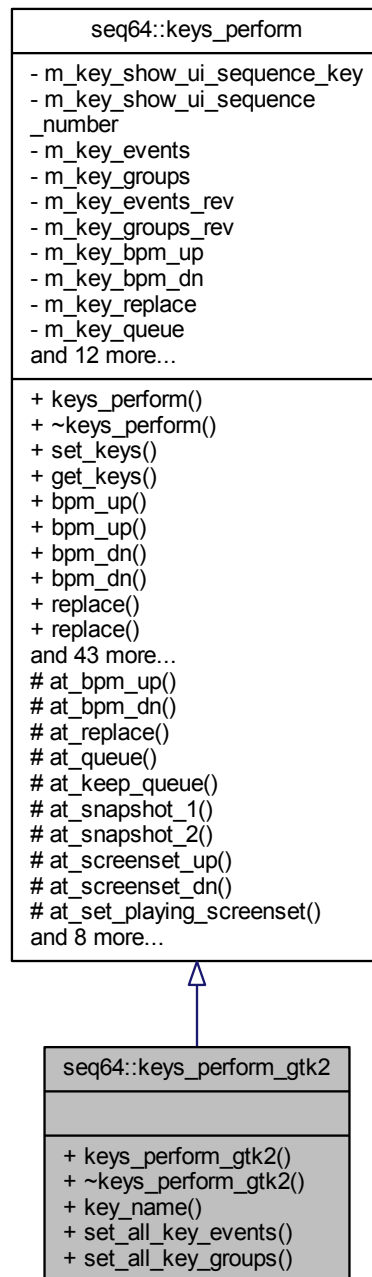
12.22.4.2 unsigned int seq64::keys_perform::m_key_bpm_up [private]

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

12.23 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

12.23.1 Detailed Description

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

12.23.2 Constructor & Destructor Documentation

12.23.2.1 `seq64::keys_perform_gtk2::~~keys_perform_gtk2 ()` [virtual]

Finally, any active patterns/sequences are deleted.

12.23.3 Member Function Documentation

12.23.3.1 `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](#).

12.23.3.2 `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](#).

12.23.3.3 `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](#).

12.24 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()`.

12.25 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
'Getter' function for member 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.

12.25.1 Detailed Description

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

12.25.2 Constructor & Destructor Documentation

- 12.25.2.1 **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.

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

Parameters

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

12.25.3 Member Function Documentation

12.25.3.1 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.

12.25.3.2 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

12.25.4 Field Documentation

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

See the SEQ64_KEYSTROKE_PRESS and SEQ64_KEYSTROKE_RELEASE readability macros.

12.25.4.2 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.

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

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

12.26 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.

12.26.1 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.

12.26.2 Constructor & Destructor Documentation

12.26.2.1 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.

12.26.3 Member Function Documentation

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

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

12.26.3.2 bool seq64::lash::process_events ()

Returns

Always returns true.

12.26.3.3 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.

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

Parameters

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

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

Currently incomplete.

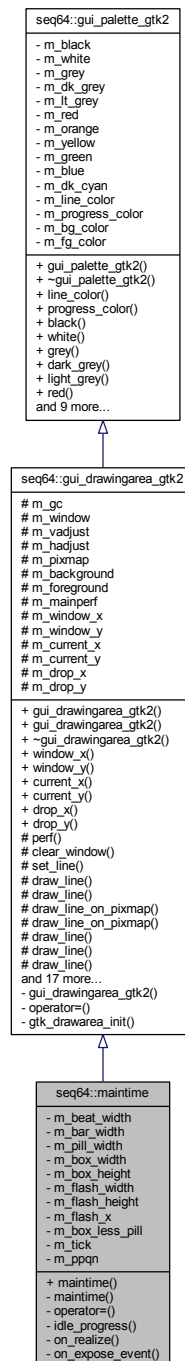
Parameters

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

12.27 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](#) ([midipulse](#) 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.
- [midipulse](#) [m_tick](#)
Saves the tick value for [on_expose_event\(\)](#).
- int [m_ppqn](#)
Provides the active PPQN value.

Additional Inherited Members

12.27.1 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.

12.27.2 Constructor & Destructor Documentation

12.27.2.1 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.

12.27.3 Member Function Documentation

12.27.3.1 `int seq64::maintime::idle_progress (midipulse 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!").

12.27.3.2 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.

12.27.3.3 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.

Actually, it might be useful after all, to avoid flickering under JACK transport. Let's put it back for now. (It doesn't help, but we will leave it in, the overhead is small.)

12.27.4 Field Documentation

12.27.4.1 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().

12.27.4.2 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().

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

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

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

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

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

Just a bit smaller than m_box_width.

12.27.4.6 `const int seq64::maintime::m_flash_height` `[private]`

Just a bit smaller than `m_box_width`.

12.27.4.7 `midipulse seq64::maintime::m_tick` `[private]`

It might actually be useful after all. And the overhead is tiny.

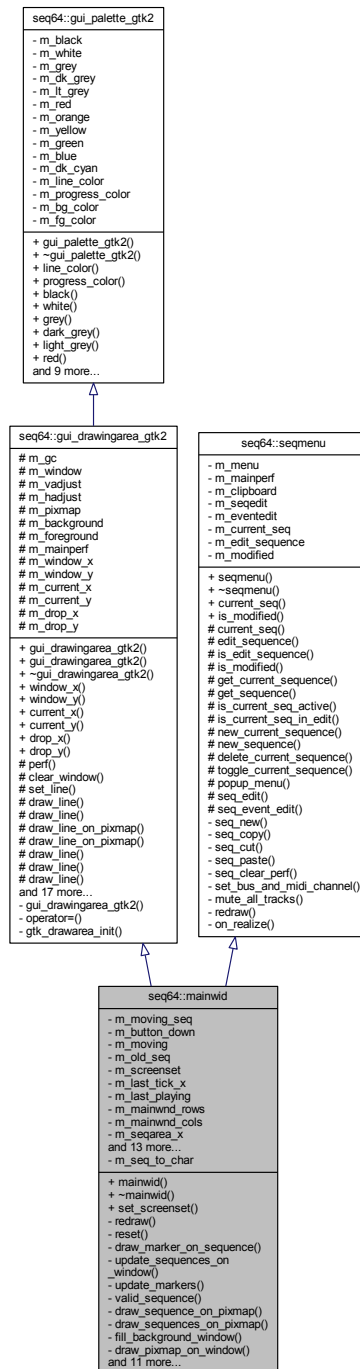
12.27.4.8 `int seq64::maintime::m_ppqn` `[private]`

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

12.28 `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 all of the members.

- [~mainwid](#) ()

A rote destructor.

- void [set_screenset](#) (int ss)

Set the current screen-set.

Private Member Functions

- virtual void [redraw](#) (int seq)
This virtual function, overridden from the seqmenu base class, draws the the given pattern/sequence again.
- void [reset](#) ()
This function redraws everything and queues up a redraw operation.
- void [draw_marker_on_sequence](#) (int seq, int tick)
Does the actual drawing of one pattern/sequence position marker, a vertical progress bar.
- void [update_sequences_on_window](#) ()
Updates the image of multiple sequencer/pattern slots.
- 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).
- 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 a sequence pixmap 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 needed.
- 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 in one of the sequence/pattern slots.
- 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

- [sequence m_moving_seq](#)
Holds a partial copy of the sequence we a moving on the patterns panel.
- [bool m_button_down](#)
Indicates that the mouse button is still down.
- [bool m_moving](#)
Indicates that we are still in the middle of a drag-and-drop operation.
- [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 or user-interface configuration items later.
- [int m_screenset_slots](#)
Provides a convenience variable for avoiding multiplications.
- [int m_screenset_offset](#)
Provides a convenience variable for avoiding multiplications.
- [int m_progress_height](#)
Provides the height of the progress bar, to save calculations and for consistency between drawing and erasing the progress bar.

Additional Inherited Members

12.28.1 Constructor & Destructor Documentation

12.28.1.1 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, motion, key presses, and focus changes.

Parameters

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

12.28.2 Member Function Documentation

12.28.2.1 void seq64::mainwid::set_screenset (`int` *ss*)

Parameters

<i>ss</i>	Provides the screen-set number to set.
-----------	--

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

Parameters

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

Implements [seq64::seqmenu](#).

12.28.2.3 `void seq64::mainwid::draw_marker_on_sequence (int seqnum, int tick) [private]`

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. Also, currently the [redraw\(\)](#) is hit when [seq_edit\(\)](#) is called, but not when [seq_event_edit\(\)](#) is called, which makes the latter not paint the in-edit highlight colors (if enabled). Why?

Parameters

<i>seqnum</i>	Provides the number of the sequence to draw.
<i>tick</i>	Provides the location to draw the marker. If pause support is compiled in (i.e. no <code>--disable-pause</code> in the configuration), then this parameter is ignored, and is replaced by the sequences' <code>get_lask_tick()</code> value. This causes correct stop/pause/play progress-bar behavior in each pattern slot.

12.28.2.4 `void seq64::mainwid::update_sequences_on_window () [private]`

Used by the friend class `mainwnd`, but also useful for our EXPERIMENTAL feature to fully highlight the current sequence.

12.28.2.5 `void seq64::mainwid::update_markers (int tick) [private]`

Parameters

<i>tick</i>	Starting point for drawing the markers.
-------------	---

12.28.2.6 `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.

12.28.2.7 `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 (muted by default) – the progress bars move in each pattern. Gaps in the sequence in the Song (performance) Editor don't change the appearance of the patterns if only the main window is up. 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. This setup holds for ALSA, but not for JACK transport.

Parameters

<i>seqnum</i>	Provides the number of the sequence slot that needs to be drawn. It is checked for validity before usage.
---------------	---

12.28.2.8 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.

12.28.2.9 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. This function is used when dragging a pattern from one pattern-slot to another pattern-slot.

We have to add 1 pixel to the y height in order to avoid leaving behind a line at the bottom of an empty pattern-slot.

Parameters

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

12.28.2.10 int seq64::mainwid::seq_from_xy (int *x*, int *y*) [private]

Parameters

<i>a</i> ↔ _x	Provides the x coordinate.
<i>a</i> ↔ _y	Provides the y coordinate.

Returns

Returns -1 if the sequence number cannot be calculated.

12.28.2.11 int seq64::mainwid::timeout () [private]

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

Returns

Always returns true.

12.28.2.12 `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.

12.28.2.13 `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 was to do this!

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

Parameters

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

Returns

Always returns true.

12.28.2.15 `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. Another way to bring up the same functionality is described in the next paragraph.

If the press is a double-click, it first acts just like two single-clicks (which might confuse the user at first, because it toggles the mute state twice). 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.

We also try to handle a Ctrl-double-click as a signal to do an event edit, instead of a sequence edit. The event editor provides a way to look at all events in detail, without having to select the type of event to see. However, this doesn't work, the event is treated like a ctrl-single-click. And we use the Alt key to enable window movement or resizing in our window manager, so that's out.

Parameters

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

Returns

Always returns true.

12.28.2.16 `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.

Also now implements the new "toggle all other patterns" action, initiated via Shift-Left-Click.

Parameters

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

Returns

Always returns true.

Try disabling the setting of the current sequence; It completely disables drag-n-drop. But leaving it in removes the current-sequence highlighting, which otherwise is fine. So we do it only if moving a pattern (drag-and-drop).

12.28.2.17 `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.

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

Just sets the Gtk::HAS_FOCUS flag.

Returns

Always returns false.

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

Just unsets the Gtk::HAS_FOCUS flag.

Returns

Always returns false.

12.28.3 Field Documentation

12.28.3.1 `sequence seq64::mainwid::m_moving_seq [private]`

The assignment is made by `sequence::partial_copy()`, which behaves like the legacy `seq24` code.

12.28.3.2 `bool seq64::mainwid::m_button_down [private]`

Used in the drag-and-drop functionality.

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

It is equal to `m_mainwnd_rows * m_mainwnd_cols`.

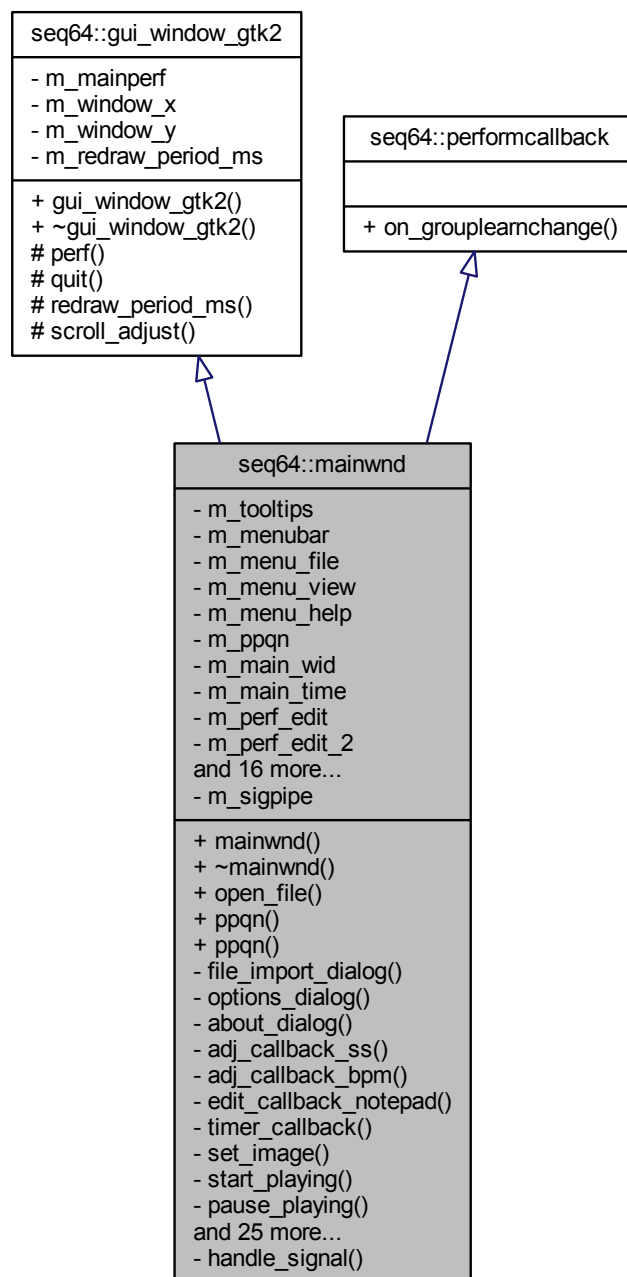
12.28.3.4 `int seq64::mainwid::m_screenset_offset [private]`

It is equally to `m_screenset_slots * m_screenset`.

12.29 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, bool allowperf2=true, int ppqn=SEQ64_USE_DEFAULT_PPQN)
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 set_image (bool isrunning)`
Changes the image used for the pause/play button.
- `void start_playing ()`
Starts playing of the song.
- `void pause_playing ()`
Pauses the playing of the song, leaving the progress bar where it stopped.
- `void stop_playing ()`
Stops the playing of the song.
- `void toggle_playing ()`
Reverses the state of playback.
- `void learn_toggle ()`
Toggle the group-learn status.
- `void open_performance_edit ()`
Opens the Performance Editor (Song Editor).
- `void open_performance_edit_2 ()`
Opens the second Performance Editor (Song Editor).
- `void enregister_perfedits ()`
This function brings together the two perfedit objects, so that they can tell each other when to queue up a draw operation.
- `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.
- int [m_ppqn](#)
Saves the PPQN value obtained from the MIDI file (or the default value, the global ppqn, if SEQ64_USE_DEFAULT_PPQN was specified in reading the MIDI file.
- 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.

- [perfedit](#) * [m_perf_edit_2](#)
A pointer to an optional second song/performance editor.
- [options](#) * [m_options](#)
A pointer to the program options.
- Gdk::Cursor [m_main_cursor](#)
Mouse cursor?
- Gtk::Image * [m_image_play](#)
Provides a pointer to hold the images for the pause/play button.
- 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::Adjustment * [m_adjust_bpm](#)
The spin/adjustment controls for the BPM (beats-per-minute) value.
- Gtk::Adjustment * [m_adjust_ss](#)
The spin/adjustment controls for the screen set value.
- Gtk::Adjustment * [m_adjust_load_offset](#)
The spin/adjustment controls for the load offset value.
- Gtk::Entry * [m_entry_notes](#)
What is this?
- bool [m_is_running](#)
Holds the current status of running, for use in display the play versus pause icon.
- sigc::connection [m_timeout_connect](#)
Provides a timeout handler.

Static Private Attributes

- static int [m_sigpipe](#) [2]
Interesting; what is this used for.

Additional Inherited Members

12.29.1 Constructor & Destructor Documentation

12.29.1.1 `seq64::mainwnd::mainwnd (perform & p, bool allowperf2 = true, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

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.
<i>allowperf2</i>	Indicates if a second perfedit window should be created. This is currently a run-time option, selectable in the "user" configuration file.

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.

View menu items 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.

12.29.2 Member Function Documentation

12.29.2.1 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.
-----------	--

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

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

12.29.2.3 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.

12.29.2.4 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.

12.29.2.5 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?

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

Let the perform object keep track of modifications.

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

Let the perform object keep track of modifications.

12.29.2.8 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. Also, if a sequence is changed by the event editor, we get a crash; need to find out how seqedit gets away with the changes.

Pause:

These do not work:

```
- midipulse tick = perf().get_jack_tick();
- midipulse tick = perf().get_max_tick();
```

12.29.2.9 void seq64::mainwnd::set_image (bool *isrunning*) [private]

Parameters

<i>isrunning</i>	If true, set the image to the "Pause" icon, since playback is running. Otherwise, set it to the "Play" button, since playback is not running.
------------------	---

12.29.2.10 void seq64::mainwnd::start_playing () [private]

The `rc_settings::jack_start_mode()` function is used (if jack is running) to determine if the playback mode is "live" (false) or "song" (true). An accessor to `perform::start_playing()`. This function is actually a callback for the pause/play button.

Note

This overrides the old behavior of playing live mode if the song is started from the main window. So let's go back to the way seq24 handles it. We could also make it dependent on the `--legacy` option, but that's too much trouble for now.

12.29.2.11 void seq64::mainwnd::pause_playing () [private]

Currently, it is just the same as `stop_playing()`, but we will get it to work.

12.29.2.12 void seq64::mainwnd::stop_playing () [private]

An accessor to perform's `stop_playing()` function. Also calls the mainwnd's `update_sequences_on_window()` function. Not sure that we need this call, since the slots seem to update anyway. But we've noticed that, with this call in place, hitting the Stop button causes a subtle change in the appearance of the first non-empty pattern of the "alofarow.mid" file.

After the Stop button is pushed (in ALSA mode), then the Space key ("start") doesn't work properly. The song starts, then quickly stops. It doesn't matter if `update_sequences_on_window()` is called or not. This happens even in seq24! This bug has proven incredibly difficult to track down, still working on it.

12.29.2.13 void seq64::mainwnd::toggle_playing () [private]

Meant only to be called when the "Play" button is pressed, if the pause feature has been compiled into the application.

12.29.2.14 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.

12.29.2.15 void seq64::mainwnd::open_performance_edit_2 () [private]

Experiment: open a second one and see what happens. It works, but one needs to tell the other to redraw if a change is made.

12.29.2.16 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.

12.29.2.17 `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.

12.29.2.18 `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`.

12.29.2.19 `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.

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

Any changed data is saved. If the pattern is playing, then it is stopped. We now use `is_running()`, instead of the global `rc().is_pattern_playing()` function.

12.29.2.21 `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.

12.29.2.22 `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.

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

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

Reimplemented from `seq64::performcallback`.

12.29.3 Field Documentation

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

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

12.29.3.2 `int seq64::mainwnd::m_ppqn` `[private]`

We need it early here to be able to pass it along to child objects.

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

The first is the Patterns Panel.

12.29.3.4 `Gtk::Button* seq64::mainwnd::m_button_play` `[private]`

If configured to support pause, it also supports the pause pixmap and functionality.

12.29.3.5 `Gtk::Adjustment* seq64::mainwnd::m_adjust_load_offset` `[private]`

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

12.30 seq64::mastermidibus Class Reference

The class that "supervises" all of the midibus objects?

Public Member Functions

- [mastermidibus](#) (int ppqn=SEQ64_USE_DEFAULT_PPQN, int bpm=c_beats_per_minute)
The mastermidibus default constructor fills the array with our busses.
- [~mastermidibus](#) ()
The destructor deletes all of the output busses, clears out the ALSA events, stops and frees the queue, and closes ALSA for this application.
- void [init](#) (int ppqn)
Initialize the mastermidibus.
- `snd_seq_t*` [get_alsa_seq](#) () const
'Getter' function for member m_alsa_seq
- int [get_num_out_buses](#) () const
'Getter' function for member m_num_out_buses
- int [get_num_in_buses](#) () const
'Getter' function for member m_num_in_buses
- void [set_beats_per_minute](#) (int bpm)
Set the BPM value (beats per minute).
- void [set_ppqn](#) (int ppqn)

- Set the PPQN value (parts per quarter note).*

 - int `get_beats_per_minute` () const
 - 'Getter' function for member m_beats_per_minute*
- int `get_ppqn` () const
 - 'Getter' function for member m_ppqn*
- std::string `get_midi_out_bus_name` (int a_bus)
 - Get the MIDI output buss name for the given (legal) buss number.*
- std::string `get_midi_in_bus_name` (int a_bus)
 - Get the MIDI input buss name for the given (legal) buss number.*
- void `print` ()
 - Print some information about the available MIDI output busses.*
- void `flush` ()
 - Flushes our local queue events out into ALSA.*
- void `start` ()
 - Starts all of the configured output busses up to m_num_out_buses.*
- void `stop` ()
 - Stops each of the output busses.*
- void `clock` (midipulse a_tick)
 - Generates the MIDI clock for each of the output busses.*
- void `continue_from` (midipulse a_tick)
 - Gets the output busses running again, if ALSA support is enabled.*
- void `init_clock` (midipulse a_tick)
 - Initializes the clock of each of the output busses.*
- int `poll_for_midi` ()
 - Initiate a poll() on the existing poll descriptors.*
- bool `is_more_input` ()
 - Test the ALSA sequencer to see if any more input is pending.*
- bool `get_midi_event` (event *a_in)
 - Grab a MIDI event.*
- void `set_sequence_input` (bool a_state, sequence *a_seq)
 - Set the input sequence object, and set the m_dumping_input value to the given state.*
- bool `is_dumping` () const
 - 'Getter' function for member m_dumping_input*
- sequence * `get_sequence` () const
 - 'Getter' function for member m_seq*
- void `sysex` (event *a_event)
 - Handle the sending of SYSEX events.*
- void `port_start` (int a_client, int a_port)
 - Start the given ALSA MIDI port.*
- void `port_exit` (int a_client, int a_port)
 - Turn off the given port for the given client.*
- void `play` (bussbyte bus, event *a_e24, midibyte a_channel)
 - Handle the playing of MIDI events on the MIDI buss given by the parameter, as long as it is a legal buss number.*
- void `set_clock` (bussbyte bus, clock_e a_clock_type)
 - Set the clock for the given (legal) buss number.*
- clock_e `get_clock` (bussbyte bus)
 - Gets the clock setting for the given (legal) buss number.*
- void `set_input` (bussbyte bus, bool a_inputing)
 - Set the status of the given input buss, if a legal buss number.*
- bool `get_input` (bussbyte bus)
 - Get the input for the given (legal) buss number.*

Private Attributes

- `snd_seq_t * m_alsa_seq`
The ALSA sequencer client handle.
- `int m_num_out_buses`
The number of output busses.
- `int m_num_in_buses`
The number of input busses.
- `midibus * m_buses_out [c_max_busses]`
Output MIDI busses.
- `midibus * m_buses_in [c_max_busses]`
Input MIDI busses.
- `midibus * m_bus_announce`
MIDI buss announcer?
- `bool m_buses_out_active [c_max_busses]`
Active output MIDI busses.
- `bool m_buses_in_active [c_max_busses]`
Active input MIDI busses.
- `bool m_buses_out_init [c_max_busses]`
Output MIDI buss initialization.
- `bool m_buses_in_init [c_max_busses]`
Input MIDI buss initialization.
- `clock_e m_init_clock [c_max_busses]`
Clock initialization.
- `bool m_init_input [c_max_busses]`
Input initialization?
- `int m_queue`
The ID of the MIDI queue.
- `int m_ppqn`
Resolution in parts per quarter note.
- `int m_beats_per_minute`
BPM (beats per minute).
- `int m_num_poll_descriptors`
The number of descriptors for polling.
- `struct pollfd * m_poll_descriptors`
Points to the list of descriptors for polling.
- `bool m_dumping_input`
For dumping MIDI input to a sequence for recording.
- `sequence * m_seq`
Points to the sequence object.
- `mutex m_mutex`
The locking mutex.

12.30.1 Constructor & Destructor Documentation

- 12.30.1.1 `seq64::mastermidibus::mastermidibus (int ppqn = SEQ64_USE_DEFAULT_PPQN, int bpm = c_beats_per_minute)`

Parameters

<i>ppqn</i>	Provides the PPQN value for this object. However, in most cases, the default, SEQ64_USE_DEFAULT_PPQN should be specified. Then the caller of this constructor should call mastermidibus::set_ppqn() to set up the proper PPQN value.
<i>bpm</i>	Provides the beats per minute value, which defaults to c_beats_per_minute.

12.30.1.2 seq64::mastermidibus::~~mastermidibus ()

Valgrind indicates we might have issues caused by the following functions:

- snd_config_hook_load()
- snd_config_update_r() via snd_seq_open()
- _dl_init() and other GNU function
- init_gtkmm_internals() [version 2.4]

12.30.2 Member Function Documentation

12.30.2.1 void seq64::mastermidibus::init (int *ppqn*)

It initializes 16 MIDI output busses, a hardwired constant, 16. Only one MIDI input buss is initialized.

12.30.2.2 void seq64::mastermidibus::set_beats_per_minute (int *bpm*)

This is done by creating an ALSA tempo structure, adding tempo information to it, and then setting the ALSA sequencer object with this information.

We fill the ALSA tempo structure (snd_seq_queue_tempo_t) with the current tempo information, set the BPM value, put it in the tempo structure, and give the tempo value to the ALSA queue.

Threadsafe

Parameters

<i>bpm</i>	Provides the beats-per-minute value to set.
------------	---

12.30.2.3 void seq64::mastermidibus::set_ppqn (int *ppqn*)

This is done by creating an ALSA tempo structure, adding tempo information to it, and then setting the ALSA sequencer object with this information. Fills the tempo structure with the current tempo information. Then sets the ppqn value. Finally, gives the tempo structure to the ALSA queue.

Threadsafe

Parameters

<i>ppqn</i>	The PPQN value to be set.
-------------	---------------------------

12.30.2.4 `std::string seq64::mastermidibus::get_midi_out_bus_name (int bus)`

Parameters

<i>bus</i>	Provides the output buss number.
------------	----------------------------------

Returns

Returns the buss name as a standard C++ string, truncated to 80-1 characters. Also contains an indication that the buss is disconnected or unconnected.

12.30.2.5 `std::string seq64::mastermidibus::get_midi_in_bus_name (int bus)`

Parameters

<i>bus</i>	Provides the input buss number.
------------	---------------------------------

Returns

Returns the buss name as a standard C++ string, truncated to 80-1 characters. Also contains an indication that the buss is disconnected or unconnected.

12.30.2.6 `void seq64::mastermidibus::flush ()`

Threadsafe

12.30.2.7 `void seq64::mastermidibus::start ()`

Threadsafe

12.30.2.8 `void seq64::mastermidibus::stop ()`

If ALSA support is enable, also drains the output, synchronizes the output queue, and then stop the queue.

Threadsafe

12.30.2.9 `void seq64::mastermidibus::clock (midipulse tick)`

Threadsafe

Parameters

<i>tick</i>	Provides the tick value with which to set the buss clock.
-------------	---

12.30.2.10 void seq64::mastermidibus::continue_from (midipulse tick)

Threadsafe

Parameters

<i>tick</i>	Provides the tick value to continue from.
-------------	---

12.30.2.11 void seq64::mastermidibus::init_clock (midipulse tick)

Threadsafe

Parameters

<i>tick</i>	Provides the tick value with which to initialize the buss clock.
-------------	--

12.30.2.12 int seq64::mastermidibus::poll_for_midi ()

Returns

Returns the result of the poll, or 0 if ALSA is not supported.

12.30.2.13 bool seq64::mastermidibus::is_more_input ()

Threadsafe

Returns

Returns true if ALSA is supported, and the returned size is greater than 0, or false otherwise.

12.30.2.14 bool seq64::mastermidibus::get_midi_event (event * inev)

Threadsafe

Parameters

<i>inev</i>	The event to be set based on the found input event.
-------------	---

12.30.2.15 void seq64::mastermidibus::set_sequence_input (bool state, sequence * seq)

Threadsafe

Parameters

<i>state</i>	Provides the dumping-input state to be set.
<i>seq</i>	Provides the sequence object to be logged as the mastermidibus's sequence. Can also be used to set a null pointer, to disable the sequence setting.

12.30.2.16 void seq64::mastermidibus::sysex (event * ev)

Threadsafe

Parameters

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

12.30.2.17 void seq64::mastermidibus::port_start (int client, int port)

Threadsafe Quite a lot is done during the lock!

Parameters

<i>client</i>	Provides the ALSA client number.
<i>port</i>	Provides the ALSA client port.

12.30.2.18 void seq64::mastermidibus::port_exit (int client, int port)

Both the input and output busses for the given client are stopped, and set to inactive.

Threadsafe

Parameters

<i>client</i>	The client to be matched and acted on.
<i>port</i>	The port to be acted on. Both parameter must be match before the buss is made inactive.

12.30.2.19 void seq64::mastermidibus::play (bussbyte bus, event * e24, midibyte channel)

Threadsafe

Parameters

<i>bus</i>	The buss to start play on.
<i>e24</i>	The seq24 event to play on the buss.
<i>channel</i>	The channel on which to play the event.

12.30.2.20 `void seq64::mastermidibus::set_clock (bussbyte bus, clock_e clocktype)`

The legality checks are a little loose, however.

Threadsafe

Parameters

<i>bus</i>	The buss to start play on.
<i>clocktype</i>	The type of clock to be set, either "off", "pos", or "mod", as noted in the midibus_common module.

12.30.2.21 `clock_e seq64::mastermidibus::get_clock (bussbyte bus)`

Parameters

<i>bus</i>	Provides the buss number to read.
------------	-----------------------------------

Returns

If the buss number is legal, and the buss is active, then its clock setting is returned. Otherwise, `e_clock_off` is returned.

12.30.2.22 `void seq64::mastermidibus::set_input (bussbyte bus, bool inputing)`

Why is another buss-count constant, and a global one at that, being used? And I thought there was only one input buss anyway!

Threadsafe

Parameters

<i>bus</i>	Provides the buss number.
------------	---------------------------

12.30.2.23 `bool seq64::mastermidibus::get_input (bussbyte bus)`

Parameters

<i>bus</i>	Provides the buss number.
------------	---------------------------

Returns

Always returns false.

12.30.3 Field Documentation

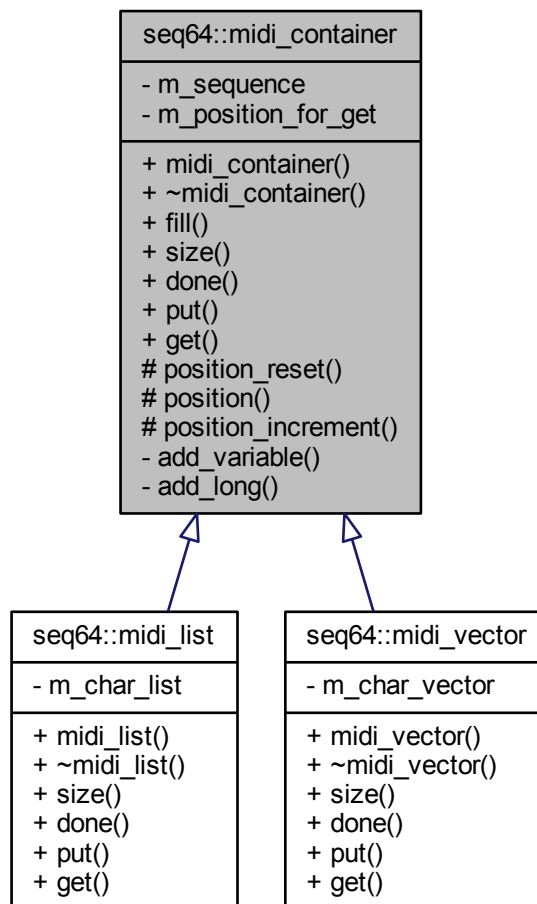
12.30.3.1 `int seq64::mastermidibus::m_beats_per_minute [private]`

We had to lengthen this name; way too easy to confuse it with "bpm" for "beats per measure".

12.31 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 (midipulse 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 (midipulse 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.

12.31.1 Member Function Documentation

12.31.1.1 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 assumed 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.
--------------------	--

12.31.1.2 `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](#).

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

It also increments `m_position_for_get`.

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

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

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

12.31.1.5 `void seq64::midi_container::add_variable (midipulse v)` `[private]`

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

12.31.1.6 `void seq64::midi_container::add_long (midipulse x)` `[private]`

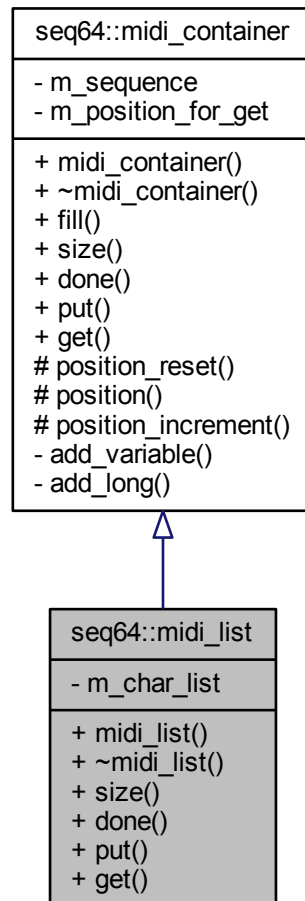
This function "replaces" `sequence::add_long_list()`.

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.

12.32 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

12.32.1 Member Typedef Documentation

12.32.1.1 `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.

12.32.2 Member Function Documentation

12.32.2.1 `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](#).

12.32.2.2 `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](#).

12.33 seq64::midi_measures Class Reference

Provides a data structure to hold the numeric equivalent of the measures string "measures:beats:divisions" ("m:b↵:d").

Public Member Functions

- [midi_measures](#) ()
Default constructor for [midi_measures](#).
- [midi_measures](#) (int [measures](#), int [beats](#), int [divisions](#))
Principal constructor for [midi_measures](#).
- int [measures](#) () const
'Getter' function for member m_measures
- void [measures](#) (int m)
'Setter' function for member m_measures We can add validation later.
- int [beats](#) () const
'Getter' function for member m_beats
- void [beats](#) (int b)
'Setter' function for member m_beats We can add validation later.
- int [divisions](#) () const
'Getter' function for member m_divisions
- void [divisions](#) (int d)
'Setter' function for member m_divisions We can add validation later.

Private Attributes

- int [m_measures](#)
The integral number of measures in the measures-based time.
- int [m_beats](#)
The integral number of beats in the measures-based time.
- int [m_divisions](#)
The integral number of divisions/pulses in the measures-based time.

12.33.1 Detailed Description

More commonly known as "bars:beats:ticks", or "BBT".

12.33.2 Field Documentation

12.33.2.1 int seq64::midi_measures::m_divisions [private]

There are two possible translations of the two bytes of a division. If the top bit of the 16 bits is 0, then the time division is in "ticks per beat" (or "pulses per quarter note"). If the top bit is 1, then the time division is in "frames per second". This function deals only with the ticks/beat definition.

12.34 seq64::midi_splitter Class Reference

This class handles the parsing and writing of MIDI files.

Public Member Functions

- [midi_splitter](#) (int [ppqn](#)=SEQ64_USE_DEFAULT_PPQN)
Principal constructor.
- [~midi_splitter](#) ()
A rote destructor.
- bool [log_main_sequence](#) ([sequence](#) &seq, int seqnum)
Logs the main sequence (an SMF 0 track) for later usage in splitting the track.
- void [initialize](#) ()
Resets the SMF 0 support variables in preparation for parsing a new MIDI file.
- void [increment](#) (int channel)
Processes a channel number by raising its flag in the `m_smf0_channels[]` array.
- bool [split](#) ([perform](#) &p, int screenset)
This function splits an SMF 0, splitting all of the channels in the sequence out into separate sequences, and adding each to the perform object.
- 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.
- int [count](#) () const
'Getter' function for member `m_smf0_channels_count`

Private Member Functions

- bool [split_channel](#) (const [sequence](#) &main_seq, [sequence](#) *seq, int channel)
This function splits the given sequence into new sequences, one for each channel found in the SMF 0 track.

Private Attributes

- 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.
- [sequence](#) * [m_smf0_main_sequence](#)
Provides support for SMF 0, points to the initial SMF 0 sequence, from which the single-channel sequences will be created.
- int [m_smf0_seq_number](#)
Provides support for SMF 0, holds the prospective sequence number of the main (SMF 0) sequence.

12.34.1 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.

12.34.2 Constructor & Destructor Documentation

12.34.2.1 seq64::midi_splitter::midi_splitter (int *ppqn* = SEQ64_USE_DEFAULT_PPQN)

Parameters

<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 SEQ64_USE_DEFAULT_PPQN, 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.
-------------	---

12.34.3 Member Function Documentation

12.34.3.1 `void seq64::midi_splitter::increment (int channel)`

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

12.34.3.2 `bool seq64::midi_splitter::split (perform & p, int screenset)`

Lastly, it adds the SMF 0 track as the last track; the user can then examine it before removing it. Is this worth the effort?

There is a little oddity, in that, if the SMF 0 track has events for only one channel, this code will still create a new sequence, as well as the main sequence. Not sure if this is worth extra code to just change the channels on the main sequence and put it into the correct track for the one channel it contains.

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. Returns false if no SMF 0 main sequence was logged.

12.34.3.3 `int seq64::midi_splitter::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 [midi_splitter](#) constructor.

```
12.34.3.4 bool seq64::midi_splitter::split_channel ( const sequence & main_seq, sequence * s, int channel )
           [private]
```

Note that the events that are read from the MIDI file have delta times. Sequencer64 converts these delta times to cumulative times. We need to preserve that here. Conversion back to delta times is needed only when saving the sequences to a file. This is done in [midi_container::fill\(\)](#).

We have to accumulate the delta times in order to be able to set the length of the sequence in pulses.

Luckily, we don't have to worry about copying triggers, since the imported SMF 0 track won't have any Seq24/↔Sequencer24 triggers.

It doesn't set the sequence number of the sequence; that is set when the sequence is added to the perform object.

Parameters

<i>main_seq</i>	This parameter is the whole SMF 0 track that was read from the MIDI file. It contains all of the channel data that needs to be split into separate sequences.
<i>s</i>	Provides the new sequence that needs to have its settings made, and all of the selected channel events added to it.
<i>channel</i>	Provides the MIDI channel number (re 0) that marks the channel data the needs to be extracted and added to the new sequence.

Returns

Returns true if at least one event got added. If none were added, the caller should delete the sequence object represented by parameter *s*.

12.34.4 Field Documentation

```
12.34.4.1 int seq64::midi_splitter::m_smf0_channels_count [private]
```

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

```
12.34.4.2 bool seq64::midi_splitter::m_smf0_channels[16] [private]
```

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

```
12.34.4.3 int seq64::midi_splitter::m_smf0_seq_number [private]
```

We want to be able to add that sequence last, for easier and cleaner removal of that sequence by the user.

12.35 seq64::midi_timing Class Reference

We anticipate the need to have a small structure holding the parameters needed to calculate MIDI times within an arbitrary song.

Public Member Functions

- [midi_timing](#) ()
Defaults constructor for [midi_timing](#).
- [midi_timing](#) (int bpmminute, int bpmmeasure, int beatwidth, int ppqn)
Principal constructor for [midi_timing](#).
- int [beats_per_minute](#) () const
'Getter' function for member m_beats_per_minute
- void [beats_per_minute](#) (int b)
'Setter' function for member m_beats_per_minute We can add validation later.
- int [beats_per_measure](#) () const
'Getter' function for member m_beats_per_measure
- void [beats_per_measure](#) (int b)
'Setter' function for member m_beats_per_measure We can add validation later.
- int [beat_width](#) () const
'Getter' function for member m_beats_per_beat_width
- void [beat_width](#) (int bw)
'Setter' function for member m_beats_per_beat_width We can add validation later.
- int [ppqn](#) () const
'Getter' function for member m_ppqn
- void [ppqn](#) (int p)
'Setter' function for member m_ppqn We can add validation later.

Private Attributes

- int [m_beats_per_minute](#)
This value should match the BPM value selected when editing the song.
- int [m_beats_per_measure](#)
This value should match the numerator value selected when editing the sequence.
- int [m_beat_width](#)
This value should match the denominator value selected when editing the sequence.
- int [m_ppqn](#)
This value provides the precision of the MIDI song.

12.35.1 Detailed Description

Although Seq24/Sequencer64 currently are heavily dependent on hard-wired values, that will be rectified eventually, so let us get ready for it.

12.35.2 Field Documentation

12.35.2.1 int seq64::midi_timing::m_beats_per_minute [private]

This value is most commonly set to 120, but is also read from the MIDI file. This value is needed if one want to calculate durations in true time units such as seconds, but is not needed to calculate the number of pulses/ticks/divisions.

12.35.2.2 int seq64::midi_timing::m_beats_per_measure [private]

This value is most commonly set to 4.

12.35.2.3 int seq64::midi_timing::m_beat_width [private]

This value is most commonly set to 4, meaning that the fundamental beat unit is the quarter note.

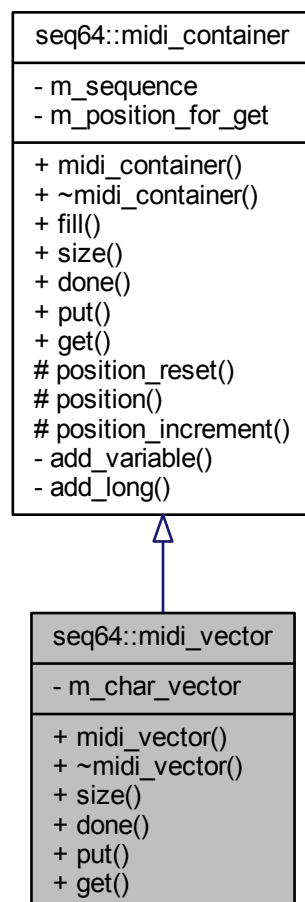
12.35.2.4 int seq64::midi_timing::m_ppqn [private]

This value is most commonly set to 192, but is also read from the MIDI file. We are still working getting "non-standard" values to work.

12.36 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

12.36.1 Member Function Documentation

12.36.1.1 `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`.

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

In this implementation, `m_position_for_get` is used.

Implements `seq64::midi_container`.

12.37 seq64::midibus Class Reference

Provides a class for handling the MIDI buss on Linux.

Public Member Functions

- [midibus](#) (int localclient, int destclient, int destport, snd_seq_t *seq, const char *client_name, const char *port_name, int id, int queue, int ppqn=SEQ64_USE_DEFAULT_PPQN)
Provides a constructor with client number, port number, ALSA sequencer support, name of client, name of port.
- [midibus](#) (int localclient, snd_seq_t *seq, int id, int queue, int ppqn=SEQ64_USE_DEFAULT_PPQN)
Secondary constructor.
- [~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 an 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](#) (midipulse tick)
Generates the MIDI clock, starting at the given tick value.
- void [continue_from](#) (midipulse tick)
Continue from the given tick.
- void [init_clock](#) (midipulse 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.
- snd_seq_t *const [m_seq](#)
ALSA sequencer client handle.
- 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.
- [midipulse](#) [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.

12.37.1 Constructor & Destructor Documentation

- 12.37.1.1 [seq64::midibus::midibus](#) (int *localclient*, int *destclient*, int *destport*, snd_seq_t * *seq*, const char * *client_name*, const char * *port_name*, int *id*, int *queue*, int *ppqn* = SEQ64_USE_DEFAULT_PPQN)

Parameters

<i>localclient</i>	Provides the local-client number.
<i>destclient</i>	Provides the destination-client number.
<i>destport</i>	Provides the destination-client port.
<i>seq</i>	Provides the sequence that will work with this buss.
<i>client_name</i>	Provides the client name, but this parameter is unused.
<i>port_name</i>	Provides the port name.
<i>id</i>	Provides the ID code for this bus. It is an index into the midibus definitions array, and is also used in the constructed human-readable buss name.
<i>queue</i>	Provides the queue ID.

12.37.1.2 `seq64::midibus::midibus (int localclient, snd_seq_t * seq, int id, int queue, int ppqn = SEQ64_USE_DEFAULT_PPQN)`

Similar to the principal constructor, but labels the buss by number more than by name.

Parameters

<i>localclient</i>	Provides the local-client number.
<i>seq</i>	Provides the sequence that will work with this buss.
<i>id</i>	Provides the ID code for this bus. It is an index into the midibus definitions array, and is also used in the constructed human-readable buss name.
<i>queue</i>	Provides the queue ID.

12.37.2 Member Function Documentation

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

Returns

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

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

Returns

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

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

Returns

Returns true, unless an error occurs.

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

Returns

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

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

Returns

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

12.37.2.6 `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.

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

Parameters

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

12.37.2.8 `void seq64::midibus::clock (midipulse tick)`

Parameters

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

12.37.2.9 `void seq64::midibus::continue_from (midipulse tick)`

Parameters

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

12.37.2.10 `void seq64::midibus::init_clock (midipulse tick)`

Parameters

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

12.37.2.11 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.
-----------------	----------------------------

12.37.3 Field Documentation

12.37.3.1 int seq64::midibus::m_clock_mod [static], [private]

Initialize this static member.

12.38 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
- bool [error_is_fatal](#) () const
'Getter' function for member m_error_is_fatal
- int [ppqn](#) () const
'Getter' function for member m_ppqn Provides a way to get the actual value of PPQN used in processing the sequences when [parse\(\)](#) was called.

Private Member Functions

- 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 into separate sequences.
- bool [parse_smf_1](#) ([perform](#) &p, int screenset, bool is_smf0=false)
This function parses an SMF 1 binary MIDI file; it is basically the original seq25 [midifile::parse\(\)](#) function.
- [midilong 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.
- int [pow2](#) (int logbase2)
Internal function for simple calculation of a power of 2 without a lot of math.
- bool [checklen](#) ([midilong](#) len, [midibyte](#) type)
Internal function to check for and report a bad length value.
- void [add_trigger](#) ([sequence](#) &seq, [midishort](#) ppqn)
Internal function to make the parser easier to read.
- [midilong read_long](#) ()
Reads 4 bytes of data using [read_byte\(\)](#).
- [midishort 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.
- [midilong read_varinum](#) ()
Read a MIDI Variable-Length Value (VLV), which has a variable number of bytes.
- void [write_long](#) ([midilong](#) value)
Writes 4 bytes, using the [write_byte\(\)](#) function.
- void [write_short](#) ([midishort](#) value)
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](#) ([midilong](#))
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](#) ([midishort](#) 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](#) ([midilong](#) tag, long len)
We want to write:
- bool [write_proprietary_track](#) ([perform](#) &a_perf)
Writes out the proprietary/SeqSpec 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_error_is_fatal](#)

Indicates if the error should be considered fatal.
- 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.
- [midi_splitter](#) [m_smf0_splitter](#)

Provides support for SMF 0.

12.38.1 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.

12.38.2 Constructor & Destructor Documentation

12.38.2.1 `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>	Provides the initial value of the PPQN setting. It is handled differently for parsing (reading) versus writing the MIDI file. <ul style="list-style-type: none"> Reading. <ul style="list-style-type: none"> If set to SEQ64_USE_DEFAULT_PPQN, the legacy application behavior is used. The <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.
<i>oldformat</i>	If true, write out the MIDI file using the old Seq24 format, instead of the new MIDI-compliant sequencer-specific format, for the seq24-specific SeqSpec tags defined in the <code>globals</code> module. This option is false by default. Note that this option is only used in writing; reading can handle either format transparently.
<i>globalbgs</i>	If true, write any non-default values of the key, scale, and background sequence to the global "proprietary" section of the MIDI file, instead of to each sequence. Note that this option is only used in writing; reading can handle either format transparently.

12.38.3 Member Function Documentation

12.38.3.1 `bool seq64::midifile::parse (perform & p, int screenset = 0)`

In addition to the standard MIDI track data in a normal track, Seq24/Sequencer64 adds four sequencer-specific events just before the end of the track:

```

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

```

Note that only Sequencer64 adds "FF 7F len" to the SeqSpec data.

Standard MIDI provides for port and channel specification meta events, but they are apparently considered obsolete:

Obsolete meta-event:	Replacement:
MIDI port (buss): FF 21 01 po	Device (port) name: FF 09 len text
MIDI channel: FF 20 01 ch	

What do other applications use for specifying port/channel?

Note on the is-modified flag: We now assume that the perform object is starting from scratch when parsing. But we let mainwnd tell the perform object when to clear everything with `perform::clear_all()`. The mainwnd does this for a new file, opening a file, but not for a file import, which might be done simply to add more MIDI tracks to the current composition. So, if parsing succeeds, all we want to do is make sure the flag is set.

Parsing a file successfully is not always a modification of the setup. For instance, the first read of a MIDI file should start clean, not dirty.

SysEx notes:

Some files (e.g. Dixie04.mid) do not always encode System Exclusive messages properly for a MIDI file. Instead of a varinum length value, they are followed by extended IDs (0x7D, 0x7E, or 0x7F).

We've covered some of those cases by disabling access to `m_data` if the position passes the size of the file, but we want try to bypass these odd cases properly. So we look ahead for one of these special values.

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. This value ranges from -31 to 0 to +31 (32 is the maximum screen-set available in Seq24). This offset is added to the sequence number read in for the sequence, to place it elsewhere in the imported tune, and locate it in a specific screen-set. If this parameter is non-zero, then we will assume that the perform data is dirty.

Returns

Returns true if the parsing succeeded. Note that the error status is saved in `m_error_is_fatal`, and a message (to display later) is saved in `m_error_message`.

12.38.3.2 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.

12.38.3.3 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.

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

The original sequence remains in place, in sequence slot 16 (the 17th slot). The user is responsible for deleting it if it is not needed.

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.

12.38.3.5 `bool seq64::midifile::parse_smf_1 (perform & p, int screenset, bool is_smf0 = false) [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.

12.38.3.6 `midilong 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 (0x24240nn), 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.

12.38.3.7 `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.

12.38.3.8 `int seq64::midifile::pow2 (int logbase2) [private]`

Use for calculating the denominator of a time signature.

Parameters

<i>logbase2</i>	Provides the power to which 2 is to be raised. This integer is probably only rarely greater than 4 (which represents a denominator of 16).
-----------------	--

Returns

Returns 2 raised to the logbase2 power.

12.38.3.9 `bool seq64::midifile::checklen (midilong len, midibyte type) [private]`

Parameters

<i>len</i>	The length value to be checked, and it should be greater than 0.
<i>type</i>	The type of meta event. Used for displaying an error.

Returns

Returns true if the length parameter is valid.

12.38.3.10 `void seq64::midifile::add_trigger (sequence & seq, midishort ppqn) [private]`

Handles only `c_triggers_new` values, not the old `c_triggers` value. If `m_ppqn` isn't set to the default value, then we must scale these triggers accordingly, just as is done for the MIDI events.

Parameters

<i>seq</i>	Provides the sequence to which the trigger is to be added.
<i>ppqn</i>	Provides the ppqn value to use to scale the tick values if <code>m_use_default_ppqn</code> is true. If 0, the ppqn value is not used.

12.38.3.11 `midilong seq64::midifile::read_long () [private]`

Warning

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

12.38.3.12 `midilong 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.

12.38.3.13 `void seq64::midifile::write_long (midilong x) [private]`

Warning

This code looks endian-dependent.

12.38.3.14 `void seq64::midifile::write_short (midishort x) [private]`

Warning

This code looks endian-dependent.

12.38.3.15 `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.

12.38.3.16 `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()`.

12.38.3.17 `void seq64::midifile::write_varinum (midilong 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.

12.38.3.18 `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.

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

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

Returns

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

12.38.3.20 `void seq64::midifile::write_seq_number (midishort 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.

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

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

Returns

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

12.38.3.22 `void seq64::midifile::write_prop_header (midilong 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/SeqSpec 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/SeqSpec 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" (SeqSpec) 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.

12.38.3.23 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_midictl 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.

```

12.38.3.24 `long seq64::midfile::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.

12.38.3.25 `long seq64::midfile::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.

12.38.3.26 `void seq64::midfile::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.

12.38.3.27 `void seq64::midfile::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.

12.38.3.28 `bool seq64::midifile::is_sysex_special_id (midibyte ch) [inline], [private]`

Parameters

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

Returns

Returns true if the byte is SysEx special ID.

12.38.4 Field Documentation

12.38.4.1 `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.

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

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

12.38.4.3 `bool seq64::midifile::m_error_is_fatal [private]`

The caller can query for this value after getting the return value from [parse\(\)](#).

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

Once is enough.

12.38.4.5 `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.

12.38.4.6 `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.

12.38.4.7 `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.

12.38.4.8 `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.

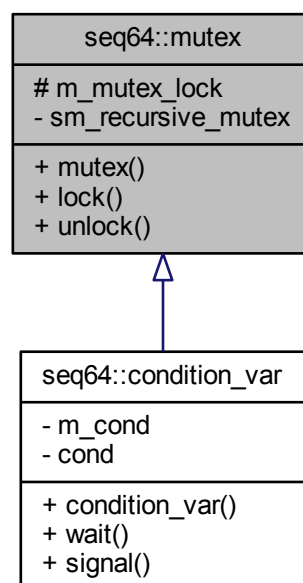
12.38.4.9 `midi_splitter seq64::midifile::m_smf0_splitter` [private]

This object holds all of the information needed to split a multi-channel sequence.

12.39 `seq64::mutex` Class Reference

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

Inheritance diagram for `seq64::mutex`:



Public Member Functions

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

Protected Attributes

- `pthread_mutex_t m_mutex_lock`
Provides a mutex lock usable by a single module or class.

Static Private Attributes

- `static const pthread_mutex_t sm_recursive_mutex`
Provides a way to disable the locking.

12.39.1 Field Documentation

12.39.1.1 `const pthread_mutex_t seq64::mutex::sm_recursive_mutex` `[static], [private]`

Define the static recursive mutex and its condition variable.

Mostly experimental, we want to disable locking to see if we can speed up MIDI file reading when the application is compiled for debugging. It takes about 8 seconds to read our sample MIDI files. This does not solve the problem of the long MIDI-file parsing, however.

```
static bool sm_mutex_enabled;
```

Provides a recursive mutex that can be used by the whole application, apparently.

12.40 seq64::options Class Reference

This class supports a full tabbed options dialog.

Inherits Dialog.

Private Types

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::CheckButton * m_button_jack_transport`
Main JACK transport selection.
- `Gtk::CheckButton * m_button_jack_master`
Main JACK transport master selection.
- `Gtk::CheckButton * m_button_jack_master_cond`
Main JACK transport master-conditional selection.
- `Gtk::Button * m_button_jack_connect`
JACK Connect button, which we need to enable/disable for clarity and some additional safety.
- `Gtk::Button * m_button_jack_disconnect`
JACK Disconnect button, which we need to enable/disable for clarity and some additional safety.
- `Gtk::Notebook * m_notebook`
Not sure yet what this notebook is for.

12.40.1 Member Enumeration Documentation

12.40.1.1 enum seq64::options::button [private]

Enumerator

e_jack_transport Defines buttons indices or IDs for some controls related to JACK. These values are handled in `options::transport_callback()`. Some of them set JACK-related values in the `rc_settings` object, while the others set up or tear down the JACK support of sequencer64.

The JACK Transport settings are a little messy. They should be radio buttons, and control each other's settings. Currently, if the user wants to set up for JACK Master, the JACK Transport button must also be checked.

Turns on the "with JACK Transport" option,
`rc_settings::with_jack_transport()`.

e_jack_master Turns on the "with JACK Master" option, `rc_settings::with_jack_master()`. If another application is already JACK Master, this will fail.

e_jack_master_cond Turns on the "with JACK Master" option `rc_settings::with_jack_master_cond()`. This option makes sequencer64 the JACK Master conditionally, that is, if no other application has claimed that role.

e_jack_start_mode_live Doesn't directly do anything; the live mode versus song mode is set by the `e_↔ jack_start_mode_song` value.

e_jack_start_mode_song Sets the "JACK start mode" value to true, which means that sequencer64 is in song mode. This value is obtained via `rc_settings::jack_start_mode()`.

e_jack_connect Causes the perform object's JACK initialization function, `perform::init_jack()`, to be called.

e_jack_disconnect Causes the perform object's JACK deinitialization function, `perform::deinit_jack()`, to be called.

12.40.2 Field Documentation

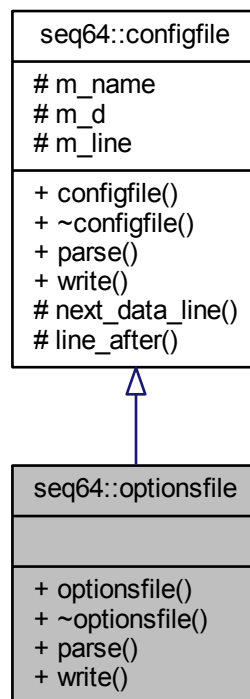
12.40.2.1 Gtk::Notebook* seq64::options::m_notebook [private]

Must be a GTK thang.

12.41 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

12.41.1 Detailed Description

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

12.41.2 Member Function Documentation

12.41.2.1 `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](#).

12.41.2.2 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.

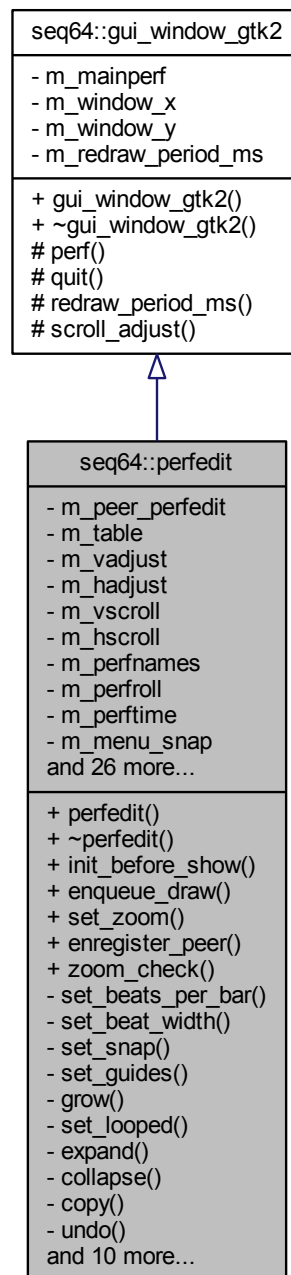
New boolean to show sequence numbers; ignored in legacy mode.

Implements [seq64::configfile](#).

12.42 seq64::perfeddit 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, bool second_perfedit=false, 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.

- void [enqueue_draw](#) (bool forward=true)
Helper wrapper for calling perfroll::queue_draw() for one or both perfedits.
- void [set_zoom](#) (int z)
Implements the horizontal zoom feature.
- void [enregister_peer](#) (perfedit *peer)
Register the peer perfedit object.

Static Public Member Functions

- static bool [zoom_check](#) (int z)
Checks zoom values for the z/Z keystrokes used in perfroll and perftime.

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 perfroll 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 [set_image](#) (bool isrunning)
Changes the image used for the pause/play button.
- void [start_playing](#) ()
Implement the playing.
- void [pause_playing](#) ()
Pauses the playing of the song, leaving the progress bar where it stopped.
- void [stop_playing](#) ()
Stop the playing.
- void [toggle_playing](#) ()

Reverses the state of playback.

- void [on_realize](#) ()

This callback function calls the base-class [on_realize\(\)](#) function, and then connects the [perfeditt::timeout\(\)](#) function to the Glib signal-timeout, with a redraw timeout of [redraw_period_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

- [perfeditt](#) * [m_peer_perfeditt](#)

The partner instance of perfedit.

- Gtk::Table * [m_table](#)

A whole horde of GUI elements.

- Gtk::Button * [m_button_play](#)

Implements the yellow two-bar pause button.

- 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.

- bool [m_is_running](#)

Holds the current status of running, for use in display the play versus pause icon.

- int [m_standard_bpm](#)

The standard "beats per measure" of Sequencer64, which here matches the beats-per-measure displayed in the perfrroll (piano roll).

Additional Inherited Members

12.42.1 Detailed Description

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

12.42.2 Constructor & Destructor Documentation

12.42.2.1 `seq64::perfeditt::perfeditt (perform & p, bool second_perfeditt = false, 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.

12.42.2.2 seq64::perfedit::~~perfedit ()

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

12.42.3 Member Function Documentation

12.42.3.1 void seq64::perfedit::init_before_show ()

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

12.42.3.2 void seq64::perfedit::enqueue_draw (bool *forward* = true)

Parameters

<i>forward</i>	If true (the default), pass the call to the peer. When passing this call to the peer, this parameter is set to false to prevent an infinite loop and the resultant stack overflow.
----------------	--

12.42.3.3 static bool seq64::perfedit::zoom_check (int *z*) [inline], [static]

It has to range from greater than 1 (the highest zoom-in causes an unexplained drawing artifact at this time), and not greater than four times the `c_perf_scale_x` value, at which point we have zoomed out so far that the measure numbers are almost completely obscured.

12.42.3.4 void seq64::perfedit::enregister_peer (perfedit * *peer*) [inline]

This function is meant to be called by mainwnd, which creates the perfedits and then makes sure they get along.

12.42.3.5 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.

12.42.3.6 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.

12.42.3.7 `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.

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

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

12.42.3.9 `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.

12.42.3.10 `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.

12.42.3.11 `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.

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

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

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

It redraws "dirty" sequences in the perffroll and the perfnames objects, and shows draw progress on the perffroll. This function is called frequently and continuously. It will work for both perfedit windows, if both are up.

12.42.3.14 `void seq64::perfedit::start_playing () [private]`

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

12.42.3.15 `void seq64::perfedit::pause_playing () [private]`

Currently, it is just the same as [stop_playing\(\)](#), but we will get it to work. Keeps the stop button enabled as a kind of rewind for ALSA.

12.42.3.16 void seq64::perfedit::toggle_playing () [inline],[private]

Meant only to be called when the "Play" button is pressed. Currently, the GUI does not change. This function will ultimately act like a Pause/Play button, but currently the pause functionality on works (partially) for JACK transport. Currently not used.

12.42.4 Field Documentation

12.42.4.1 int seq64::perfedit::m_bpm [private]

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

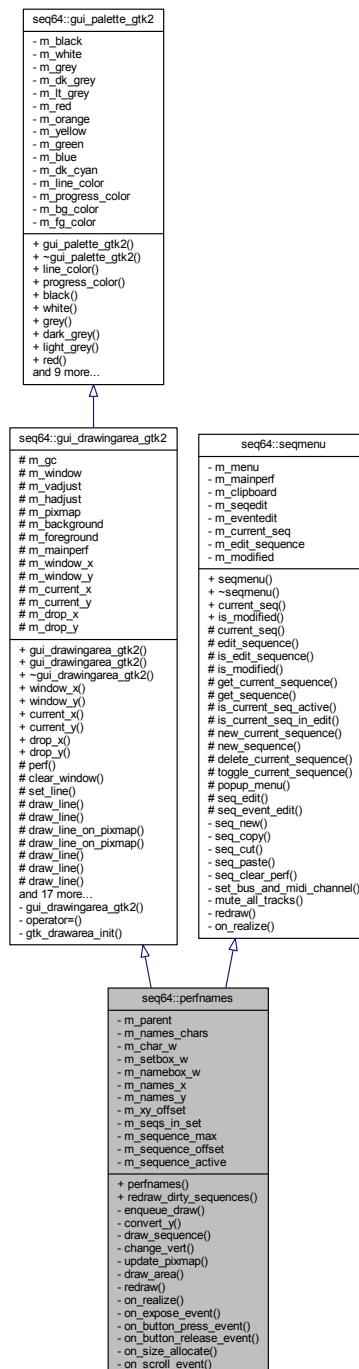
12.42.4.2 int seq64::perfedit::m_bw [private]

The denominator of the time signature.

12.43 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, [perfed](#) &parent, [Gtk::Adjustment](#) &vadjust)

Principal constructor for this user-interface object.

- void [redraw_dirty_sequences](#) ()

Redraws sequences that have been modified.

Private Member Functions

- void [enqueue_draw](#) ()
Wraps queue_draw() and forwards the call to the parent perfedit, so that it can forward it to any other perfedit that exists.
- 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

- [perfedit](#) & [m_parent](#)
Provides a link to the perfedit that created this object.
- 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

12.43.1 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).

12.43.2 Constructor & Destructor Documentation

12.43.2.1 `seq64::perfnames::perfnames (perform & p, perfedit & parent, 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.

12.43.3 Member Function Documentation

12.43.3.1 `void seq64::perfnames::enqueue_draw () [private]`

The parent perfedit will call `perfnames::queue_draw()` on behalf of this object, and it will pass a [perfnames←::enqueue_draw\(\)](#) to the peer perfedit's perfnames, if the peer exists.

12.43.3.2 `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.

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

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

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

It draws all of the sequences.

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

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

12.43.4 Field Documentation

12.43.4.1 `perfedit& seq64::perfnames::m_parent [private]`

We want to support two perfedit windows, but the children of perfedit will have to communicate changes requiring a redraw through the parent.

12.43.4.2 `int seq64::perfnames::m_names_chars [private]`

Pretty much hardwired to 24 at present.

12.43.4.3 `int seq64::perfnames::m_char_w [private]`

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

12.43.4.4 `int seq64::perfnames::m_setbox_w [private]`

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

12.43.4.5 `int seq64::perfnames::m_namebox_w [private]`

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

12.43.4.6 `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.

12.43.4.7 `int seq64::perfnames::m_xy_offset [private]`

Currently hardwired.

12.44 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`
- `int get_beats_per_bar () const`
- 'Getter' function for member `m_beats_per_bar`
- `void set_beats_per_bar (int bpm)`
- 'Setter' function for member `m_beats_per_bar`
- `int get_beat_width () const`
- 'Getter' function for member `m_beat_width`
- `void set_beat_width (int bw)`
- 'Setter' function for member `m_beat_width`
- `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_jack_running () const`
- 'Getter' function for member `m_jack_asst.is_running()` This function is useful for announcing the status of JACK in user-interface items that only have access to the perform object.
- `bool is_paused () const`
- 'Getter' function for member `m_is_paused`
- `bool is_pausable () const`
- 'Getter' function for member `m_is_paused` and `! m_jack_asst.is_running()` We might just make this internal.
- `bool is_learn_mode () const`
- 'Getter' function for member `m_mode_group_learn`
- `void enregister (performcallback *pfc)`
- Adds a pointer to an object to be notified by this perform object.
- `void clear_all ()`
- Clears all of the patterns/sequences.
- `void launch (int ppqn)`
- Calls the MIDI buss and JACK initialization functions and the input/output thread-launching functions.
- `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.
- `void print_triggers () const`
- Shows all the triggers of all the sequences.
- `void finish ()`
- The rough opposite of `launch()`; it doesn't stop the threads.
- `midipulse get_tick () const`

- 'Getter' function for member m_tick*
- `midipulse get_jack_tick () const`
- 'Getter' function for member m_jack_tick*
- `void set_jack_tick (midipulse tick)`
- 'Setter' function for member m_jack_tick*
- `void set_left_tick (midipulse tick, bool setstart=true)`
- Set the left marker at the given tick.*
- `midipulse get_left_tick () const`
- 'Getter' function for member m_left_tick*
- `void set_start_tick (midipulse tick)`
- 'Setter' function for member m_starting_tick*
- `midipulse get_max_tick () const`
- Gets the max-tick value of all active sequences.*
- `void set_right_tick (midipulse tick, bool setstart=true)`
- Set the right marker at the given tick.*
- `midipulse 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, midipulse tick)`
- Convenience function for perfroll's split-trigger functionality.*
- `midipulse get_max_trigger ()`
- Locates the largest trigger value among the active sequences.*
- `void collapse ()`
- Convenience function for perfedit's collapse functionality.*
- `void copy ()`
- Convenience function for perfedit's copy functionality.*
- `void expand ()`
- Convenience function for perfedit'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) const
- Checks the pattern/sequence for activity.*

 - [sequence](#) * [get_sequence](#) (int seq)
- Retrieves the actual sequence, based on the pattern/sequence number.*

 - void [reset_sequences](#) (bool pause=false)

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](#) (midipulse tick)

Plays all notes to the current tick.
- void [set_orig_ticks](#) (midipulse tick)

For every pattern/sequence that is active, sets the "original tick" 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.
- std::string [key_name](#) (unsigned int k) const

Here follows a few forwarding functions for the keys_perform-derived classes.
- 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\(\)](#).

- unsigned int `lookup_keyevent_key` (long seqnum)
Getters of keyboard mapping for sequence and groups.
- void `start_playing` (bool songmode=false)
'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 `pause_playing` ()
Pause playback, so that progress bars stay where they are, and playback always resumes where it left off, even in ALSA mode.
- void `stop_playing` ()
Encapsulates a series of calls used in `mainwnd`.
- void `start_key` (bool songmode=false)
Invoke the start key functionality.
- void `pause_key` (bool songmode=false)
Invoke the pause key functionality.
- void `stop_key` ()
Invoke the stop key functionality.
- 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.
- bool `is_smf_0` (const `sequence` &seq) const
True if the sequence is an SMF 0 sequence.
- 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.
- bool `playback_key_event` (const `keystroke` &k, bool songmode=false)
New function provided to unify the stop/start (space/escape) behavior of the various windows where playback can be started, paused, or stopped.

Private Member Functions

- void [launch_input_thread](#) ()
Creates the input thread using [input_thread_func\(\)](#).
- void [launch_output_thread](#) ()
Creates the output thread using [output_thread_func\(\)](#).
- bool [init_jack](#) ()
Initializes JACK support, if SEQ64_JACK_SUPPORT is defined.
- bool [deinit_jack](#) ()
Tears down the JACK infrastructure.
- 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](#)
- 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 sets [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_beats_per_bar](#)
Holds the beats/bar value as obtained from the MIDI file.
- int [m_beat_width](#)
Holds the beat width value as obtained from the MIDI file.
- [midipulse](#) [m_one_measure](#)
*Holds the "one measure's worth" of pulses (ticks), which is normally $m_ppqn * 4$.*
- [midipulse](#) [m_left_tick](#)
Holds the position of the left (L) marker, and it is first defined as 0.
- [midipulse](#) [m_right_tick](#)
Holds the position of the right (R) marker, and it is first defined as the end of the fourth measure.
- [midipulse](#) [m_starting_tick](#)
Holds the starting tick for playing.
- [midipulse](#) [m_tick](#)
MIDI Clock support.
- [midipulse](#) [m_jack_tick](#)
Let's try to save the last JACK pad structure tick for re-use with resume after pausing.
- bool [m_usemidiclock](#)
More MIDI clock support.
- bool [m_midiclockrunning](#)
More MIDI clock support.
- int [m_midiclocktick](#)
More MIDI clock support.
- int [m_midiclockpos](#)
More MIDI clock support.
- bool [m_is_paused](#)
Support for pause, which does not reset the "last tick" when playback stops/starts.
- 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 indices.

Friends

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

12.44.1 Detailed Description

It has way too many data members, one of them public. Might be ripe for refactoring. That has its own dangers, of course.

12.44.2 Constructor & Destructor Documentation

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

Also note that we have a little issue with the fact that various sequences (patterns) can potentially have different beats/measure and beat-width values.

Currently, when reading the MIDI file, the beats/minute value is obtained from the MIDI file, if present, and this value is passed to `perform::set_beats_per_minute()`, which forwards it to the master MIDI buss and JACK assistant objects. This Tempo setting comes from both the Tempo meta event in track 0, and from the Seq24's `c_bpm` SeqSpec section! This setting is now also made for the two Time Signature values.

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 <code>gui_assistant_gtk2</code> class in the <code>seq_gtkmm2</code> GUI-specific library. Note that we access the <code>m_gui_support</code> member using the <code>gui()</code> accessor function.
<code>ppqn</code>	The default, choosable, or actual PPQN value.

12.44.2.2 seq64::perform::~~perform ()

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

12.44.3 Member Function Documentation

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

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

12.44.3.2 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.

12.44.3.3 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.

12.44.3.4 void seq64::perform::launch (int *ppqn*)

This function is called in main(). We collected all the calls here as a simplification, and renamed it because it is more than just initialization. This function must be called after the perform constructor and after the configuration file and command-line configuration overrides.

Parameters

<i>ppqn</i>	Provides the PPQN value, which is either the default value (192) or is read from the "user" configuration file.
-------------	---

12.44.3.5 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 mainwid 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.
------------	--

12.44.3.6 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.

12.44.3.7 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?

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

Parameters

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

12.44.3.9 void seq64::perform::finish () [inline]

A minor simplification for the main() routine, hides the JACK support macro.

12.44.3.10 `void seq64::perform::set_left_tick (midipulse tick, bool setstart = true)`

We let the caller determine if this setting is a modification. If the left tick is later than the right tick, the right tick is move to one measure past the left tick.

Todo The `perform::m_one_measure` member is currently hardwired to `PPQN * 4`.

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" (<code>m_ppqn * 4</code>) 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.

12.44.3.11 `midipulse seq64::perform::get_max_tick () const`

We can't seem to find a way to get a good value from the perform object itself, so we get it from the sequences. This approach kind of works, but the result is slow. We could optimize it a little by saving the indices of the first and last active sequence.

Returns

If no active sequence is found, 0 is returned. Otherwise, the maximum `get_last_tick()` value of the active sequences is returned.

12.44.3.12 `void seq64::perform::set_right_tick (midipulse 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 is 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" (<code>m_ppqn * 4</code>) 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.

12.44.3.13 `void seq64::perform::move_triggers (bool direction)`

Parameters

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

12.44.3.14 `void seq64::perform::copy_triggers ()`

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

12.44.3.15 `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.

12.44.3.16 `midipulse 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?

12.44.3.17 `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.
------------	---

12.44.3.18 `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.
------------	---

12.44.3.19 `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.
------------	---

12.44.3.20 `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 the following values:

```
c_midi_control_mod_replace
c_midi_control_mod_snapshot
c_midi_control_mod_queue
c_midi_control_mod_gmute
c_midi_control_mod_glearn
```

12.44.3.21 `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.

12.44.3.22 `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.

12.44.3.23 `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, so we don't.

Parameters

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

12.44.3.24 `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.

12.44.3.25 `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 <code>m_seqs_in_set-1</code> .
-----------------	--

12.44.3.26 `void seq64::perform::unset_mode_group_learn ()`

Then unsets the group-learn mode flag.

12.44.3.27 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.
----------------	---

12.44.3.28 void seq64::perform::start (bool *state*)

Parameters

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

12.44.3.29 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`?

12.44.3.30 void seq64::perform::position_jack (bool *state*)

If we run "klik -j -P" and then start Sequencer64, we get:

```
klick: src/metronome_jack.cc:52: virtual void
MetronomeJack::process_callback(sample_t*, nframes_t): Assertion
'pos.beat > 0 && pos.beat <= pos.beats_per_bar' failed.
```

Let's try enabling the relocate parameter if we're JACK Master. This removes the error from klick, but it clicks at an extremely fast rate!

12.44.3.31 void seq64::perform::all_notes_off ()

Then flush the MIDI buss.

12.44.3.32 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.

12.44.3.33 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.
------------	---

12.44.3.34 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.

12.44.3.35 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.

12.44.3.36 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.

12.44.3.37 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.

12.44.3.38 `bool seq64::perform::is_active (int seq) const` `[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.

12.44.3.39 `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.

12.44.3.40 `void seq64::perform::reset_sequences (bool pause = false)`

Note that these calls could be folded into one member function of the sequence class. [Done!]

Finally, flush the MIDI buss.

12.44.3.41 `void seq64::perform::play (midipulse 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.
-------------	--

12.44.3.42 `void seq64::perform::set_orig_ticks (midipulse tick)`

This is really the "last tick" value, so we renamed `sequence::set_orig_tick()` to `sequence::set_last_tick()`.

Parameters

<i>tick</i>	Provides the last-tick value to be set for each sequence that is active.
-------------	--

12.44.3.43 `void seq64::perform::set_beats_per_minute (int bpm)`

Replaces `perform::set_bpm()` from `seq24`.

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. This setting does get saved to the MIDI file, with the `c_bpmtag`.

12.44.3.44 `void seq64::perform::set_sequence_control_status (int status)`

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

12.44.3.45 `void seq64::perform::unset_sequence_control_status (int status)`

Then the given status is reversed in `m_control_status`.

12.44.3.46 `void seq64::perform::sequence_playing_on (int seq)`

Parameters

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

12.44.3.47 `void seq64::perform::sequence_playing_off (int seq)`

Parameters

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

12.44.3.48 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.

12.44.3.49 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 <code>m_mute_group_selected * m_seqs_in_set</code> .
---------------	--

Returns

Returns the value of `m_mute_group[gtrack + set offset]`

12.44.3.50 void seq64::perform::output_func ()

This function is called by the free function [output_thread_func\(\)](#). Here's how it works:

- It runs while `m_outputting` is true.
- MORE TO COME

Change Note ca 2016-01-26 Hurray, seq24 is coming back to life! We see that there is a fix for clock tick drift here, which relies on using long and long long values. See the Changelog for seq24 0.9.3.

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.

12.44.3.51 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.
---------------	---------------------

12.44.3.52 `void seq64::perform::save_playing_state ()`

Inactive patterns get the value set to false.

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

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

12.44.3.54 `bool seq64::perform::show_ui_sequence_number () const [inline]`

Used in mainwid, optionsfile, and perform.

12.44.3.55 `unsigned int seq64::perform::lookup_keyevent_key (long seqnum) [inline]`

If not found, returns something "safe" [so use `get_key()->count()` to see if it's there first]

12.44.3.56 `void seq64::perform::start_playing (bool songmode = false)`

Encapsulates a series of calls used in mainwnd.

Actually, we can use the `m_running` variable.

```
bool is_playing () const { return rc().is_pattern_playing(); // Change Note ca 2016-03-19 return m_running; }
```

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.

Note that the complementary function, `stop_playing()`, is an inline function defined in the header file.

Note

It would be nice to know why the following code snippet disables the mute/unmute functionality of the performance/song editor:

```
position_jack(false);
start_jack();
start(false);
```

The `jack_assistant::position()` function doesn't use the boolean parameter at present; that code is effectively disabled.

Okay, now it does, if the "relocate" parameter is true. See `perform::position_jack()` and `jack_assistant::position()`. This parameter, when true, allows the "klick" application to get proper position data.

The `perform::start()` function passes its boolean flag to `perform::inner_start()`, which sets the playback mode to that flag; if that flag is false, that turns off "song" mode. So that explains why mute/unmute is disabled.

- `seqroll` and `mainwnd`: `position_jack(false); start(false); start_jack()`.
- `perfedit`: `position_jack(true); start_jack(); start(true)`.

Parameters

<i>songmode</i>	Indicates if the caller wants to start the playback in JACK mode. In the seq42 (yes, "42", not "24") code at GitHub, this flag was identical to the "global_jack_start_mode" flag, which is true for Song mode, and false for Live mode. False disables Song mode, and is the default, which matches seq24. If the previous state was "paused", then we start it in Live mode, which works because Song mode has already turned on the sequences. This method is not quite intuitive, because it is really following Live mode.
-----------------	---

12.44.3.57 void seq64::perform::pause_playing ()

Currently almost the same as [stop_playing\(\)](#), but expanded as noted in the comments so that we ultimately have more granular control over what happens. We're researching the whole sequence of stopping and starting, and it can be tricky to make correct changes.

We still need to make restarting pick up at the same place in ALSA mode; in JACK mode, JACK transport takes care of that feature.

12.44.3.58 void seq64::perform::stop_playing ()

Stops playback, turns off the (new) `m_is_paused` flag, and set the "is-pattern-playing" flag to false. Do we need to reset the `m_running` flag here?

12.44.3.59 void seq64::perform::start_key (bool *songmode* = false)

Meant to be used by GUIs to unify the treatment of keys versus buttons.

Parameters

<i>songmode</i>	The live/play mode parameter to be passed along to the key processor.
-----------------	---

12.44.3.60 void seq64::perform::pause_key (bool *songmode* = false)

Meant to be used by GUIs to unify the treatment of keys versus buttons.

Parameters

<i>songmode</i>	The live/play mode parameter to be passed along to the key processor, when starting playback.
-----------------	---

12.44.3.61 void seq64::perform::stop_key ()

Meant to be used by GUIs to unify the treatment of keys versus buttons.

12.44.3.62 `int seq64::perform::decrement_beats_per_minute () [inline]`

Actually does a lot of work in those function calls.

12.44.3.63 `int seq64::perform::increment_beats_per_minute () [inline]`

Actually does a lot of work in those function calls.

12.44.3.64 `bool seq64::perform::highlight (const sequence & seq) const [inline]`

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

12.44.3.65 `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, unless it is an SMF 0 null channel (0xFF), in which case it is 0.

Note

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

Parameters

<code>seq</code>	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.

12.44.3.66 `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_LABELS_ON_SEQUENCE) that causes all the sequences to be dirtied, and thus get redrawn with the new user-interface setting.
<i>active</i>	Indicates whether the buss or the user-interface feature is active or inactive.

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

Returns

Returns true if the key was handled.

12.44.3.68 `bool seq64::perform::perroll_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.

12.44.3.69 `bool seq64::perform::playback_key_event (const keystroke & k, bool songmode = false)`

To be used in mainwnd, perfedit, and seqroll.

The start/end key may be the same key (e.g. Space) to allow toggling when the same key is mapped to both triggers.

Checking [is_running\(\)](#) may not work completely in JACK.

Parameters

<i>k</i>	Provides the encapsulated keystroke to check.
<i>songmode</i>	Provides the "jack flag" needed by the mainwnd, seqroll, and perfedit windows. Defaults to false, which disables Song mode, and enables Live mode.

Returns

Returns true if the keystroke matched the start, stop, or (new) pause keystrokes.

12.44.3.70 `void seq64::perform::launch_input_thread () [private]`

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

12.44.3.71 `void seq64::perform::launch_output_thread () [private]`

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

12.44.3.72 `bool seq64::perform::init_jack () [inline],[private]`

Who calls this routine? The main() routine of the application [via [launch\(\)](#)], and the options module, when the Connect button is pressed.

Returns

Returns the result of the init() call; true if JACK sync is now running. If JACK support is not built into the application, then this function returns false, to indicate that JACK is (definitely) not running.

12.44.3.73 `bool seq64::perform::deinit_jack () [inline],[private]`

Called by [launch\(\)](#) and in the options module, when the Disconnect button is pressed.

Returns

Returns the result of the init() call; false if JACK sync is now no longer running. If JACK support is not built into the application, then this function returns true, to indicate that JACK is (definitely) not running.

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

Parameters

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

Returns

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

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

Parameters

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

Returns

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

12.44.3.76 `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.

12.44.3.77 `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 [0, <code>m_sequence_max</code>).
------------	--

Returns

Returns true if the sequence number is valid.

12.44.3.78 `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.

12.44.3.79 `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 added. 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.

12.44.3.80 `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.

Minor issue:

In ALSA mode, restarting the sequence moves the progress bar to the beginning of the sequence, even if just pausing.

Parameters

<i>state</i>	Sets the playback mode, and, if true, turns off all of the sequences.
--------------	---

12.44.3.81 `void seq64::perform::inner_stop () [private]`

Note that we do need to set the running flag to false here, even when JACK is running. Otherwise, JACK starts ping-ponging back and forth between positions under some circumstances.

However, if JACK is running, we do not want to reset the sequences... this causes the progress bar for each sequence to remove to near the end of the sequence.

12.44.3.82 `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!

12.44.3.83 `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.

12.44.3.84 `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.

12.44.4 Friends And Related Function Documentation

12.44.4.1 `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.

The transport state will be:

- `JackTransportStopped` when a new position is requested.
- `JackTransportStarting` when the transport is waiting to start.
- `JackTransportRolling` when the timeout has expired, and the position is now a moving target.

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.

Returns

Returns 1 if the function works, and 0 if something was wrong.

12.44.5 Field Documentation

12.44.5.1 `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.

12.44.5.2 `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.

12.44.5.3 `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.

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

There are two, "live" and "song", indicated by the following values:

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

12.44.5.5 `int seq64::perform::m_beats_per_bar` `[private]`

The default value is SEQ64_DEFAULT_BEATS_PER_MEASURE (4).

12.44.5.6 `int seq64::perform::m_beat_width` `[private]`

The default value is SEQ64_DEFAULT_BEAT_WIDTH (4).

12.44.5.7 `midipulse 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.

12.44.5.8 `midipulse seq64::perform::m_left_tick` `[private]`

Note that "tick" is actually "pulses".

12.44.5.9 `midipulse seq64::perform::m_right_tick` `[private]`

Note that "tick" is actually "pulses".

12.44.5.10 `midipulse 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".

12.44.5.11 `midipulse seq64::perform::m_tick` `[mutable], [private]`

The `m_tick` member holds the tick to be used in displaying the progress bars and the maintime pill. It is mutable because sometimes we want to adjust it in a const function for pause functionality.

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

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

12.44.5.13 `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.

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

Used by the [install_sequence\(\)](#) function. Note that this value is not a suitable replacement for `c_max_sequence/m←_sequence_max`, because there can be inactive sequences amidst the active sequences.

12.44.5.15 `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?

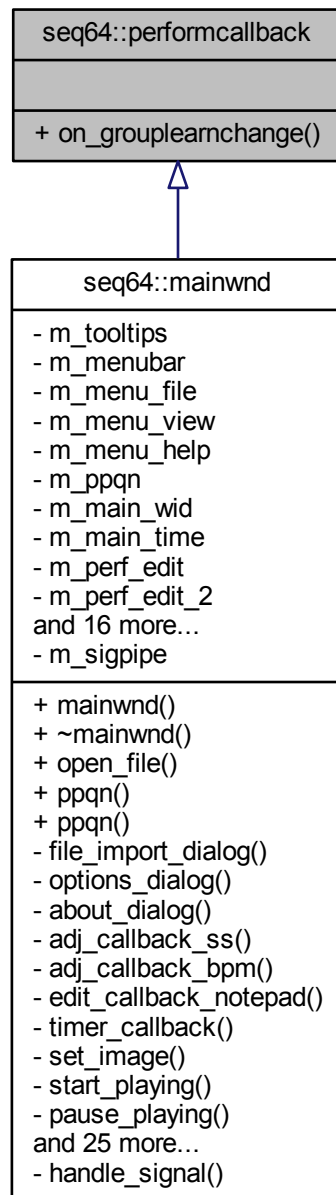
12.44.5.16 `bool seq64::perform::m_is_modified` `[private]`

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

12.45 seq64::performcallback Struct Reference

Provides for notification of events.

Inheritance diagram for seq64::performcallback:



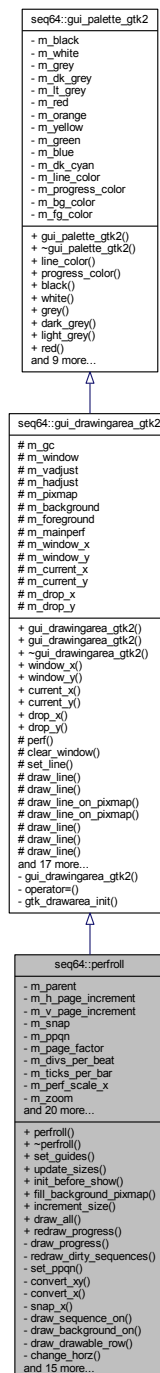
12.45.1 Detailed Description

Provide a response to a group-learn change event.

12.46 seq64::perfrroll Class Reference

This class implements the performance roll user interface.

Inheritance diagram for seq64::perfroll:



Public Member Functions

- [perfroll](#) ([perform](#) &[perf](#), [perfedit](#) &parent, Gtk::Adjustment &hadjust, Gtk::Adjustment &vadjust, int ppqn=S↵EQ64_USE_DEFAULT_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 piano roll.
- void [increment_size](#) ()
*Increments the value of `m_roll_length_ticks` by the `PPQN * 512`, then calls [update_sizes\(\)](#).*
- void [draw_all](#) ()
Provides a very common sequence of calls used in `perfroll_input`.
- void [redraw_progress](#) ()
Helper function to simplify the client call.

Private Member Functions

- 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 [set_ppqn](#) (int ppqn)
Handles changes to the PPQN value in one place.
- void [convert_xy](#) (int x, int y, [midipulse](#) &ticks, int &seq)
Converts (x, y) coordinates on the piano roll to tick (pulse) and sequence numbers.
- void [convert_x](#) (int x, [midipulse](#) &ticks)
Converts a tick-offset on the x coordinate.
- void [snap_x](#) (int &x)
This function performs a 'snap' action on x.
- 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](#), [midipulse](#) tick)
Splits a trigger, whatever that means.
- void [enqueue_draw](#) ()
Wraps `queue_draw()` and forwards the call to the parent `perfedit`, so that it can forward it to any other `perfedit` that exists.
- void [set_zoom](#) (int z)
Implements the horizontal zoom feature.
- 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

- [perfedit](#) & [m_parent](#)
Provides a link to the perfedit that created this object.
- int [m_h_page_increment](#)
Provides the horizontal page increment for the horizontal scrollbar.
- int [m_v_page_increment](#)
Provides the vertical page increment for the vertical scrollbar.
- int [m_zoom](#)
New value to attempt a rudimentary time-zoom feature.
- [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

12.46.1 Constructor & Destructor Documentation

12.46.1.1 seq64::perfroll::~~perfroll ()

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

12.46.2 Member Function Documentation

12.46.2.1 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

<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 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() .

12.46.2.2 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.

12.46.2.3 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.

12.46.2.4 void seq64::perfroll::fill_background_pixmap ()

The first thing done is to clear the background by painting it with a filled white rectangle.

This function is called whenever something occurs (e.g. zoom) that can affect how the piano roll is drawn.

12.46.2.5 void seq64::perfroll::draw_progress () [private]

We would like to be able to leave the line there when the progress is paused while running off of JACK transport. How? The `perf().get_tick()` call always returns 0 when stop is in force.

If we comment out the erasure of the old line, we see that the progress bar is also erased when a pattern boundary is hit (triggers), and when the sequence is stopped by the user.

In order to support true pause in the song editor, we tried to replace `perform::get_tick()` with `perform::get_start_tick()` and `perform::get_last_tick()` [a new experimental function]. But those replacements here always return 0, even as `perform::get_tick()` increases. Now were are trying a newer function, `perform::get_max_tick()`, which seems to do the trick for resuming (instead of rewinding) the progress bar. It's still a tiny bit laggy, so we have to find a faster way to get the maximum. (Note that the `draw_progress` function is called at every timeout, that is, constantly.)

12.46.2.6 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.

12.46.2.7 void seq64::perfroll::convert_xy (int x, int y, midipulse & d_tick, int & d_seq) [private]

The results are returned via the `d_tick` and `d_seq` parameters. The sequence number is clipped to a legal value (0 to `m_sequence_max`).

Parameters

	<i>x</i>	The x coordinate of the mouse pointer.
	<i>y</i>	The y coordinate of the mouse pointer.
out	<i>d_tick</i>	Holds the calculated tick value.
out	<i>d_seq</i>	Holds the calculated sequence-number value.

12.46.2.8 void seq64::perfroll::convert_x (int x, midipulse & tick) [private]

The result is returned via the `tick` parameter.

12.46.2.9 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.

12.46.2.10 `void seq64::perfroll::draw_sequence_on (int seqnum) [private]`

Statement nesting from hell!

12.46.2.11 `void seq64::perfroll::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.

12.46.2.12 `void seq64::perfroll::enqueue_draw () [private]`

The parent *perfed*it will call *perfroll::queue_draw()* on behalf of this object, and it will pass a [perfroll::enqueue_draw\(\)](#) to the peer *perfed*it's *perfroll*, if the peer exists.

12.46.2.13 `void seq64::perfroll::set_zoom (int z) [private]`

ca 2016-04-05 The initial zoom value is *c_perf_scale_x* (32). We allow it to range from 1 to 128, for now. Smaller values zoom in.

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

Calls the base-class version first.

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

12.46.2.15 `bool seq64::perfroll::on_expose_event (GdkEventExpose * ev) [private]`

Returns

Always returns true.

12.46.2.16 `bool seq64::perfroll::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.

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

This gives us Seq24 versus Fruity behavior.

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

If we don't check the event type first, then the `ev->keyval` value is something weird like 65507. Note that we pass the functionality on to the `perform::perfroll_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 `perfroll_key_event()` function. The `perfroll_key_event()` call handles Del, Ctrl-X, Ctrl-C, Ctrl-V, and Ctrl-Z (which does nothing at present).

Note that, even though we filter out the Ctrl key here, it still works for Ctrl-X (cut) and Ctrl-V (paste). For undo, the Undo button can be used, Ctrl-Z never worked in this view anyway.

12.46.3 Friends And Related Function Documentation

12.46.3.1 `friend class FruityPerfInput [friend]`

The `perfedit` class needs access to the private `enqueue_draw()` function.

12.46.4 Field Documentation

12.46.4.1 `perfedit& seq64::perfroll::m_parent [private]`

We want to support two `perfedit` windows, but the children of `perfedit` will have to communicate changes requiring a redraw through the parent.

12.46.4.2 `int seq64::perfroll::m_h_page_increment [private]`

It was set to 1, the same as the step increment. That is too little. This value will be set to 4, for now. Might be a useful "user" configuration option.

12.46.4.3 `int seq64::perfroll::m_v_page_increment [private]`

It was set to 1, the same as the step increment. That is too little. This value will be set to 8, for now. Might be a useful "user" configuration option.

12.46.4.4 `FruityPerfInput seq64::perfroll::m_fruity_interaction [private]`

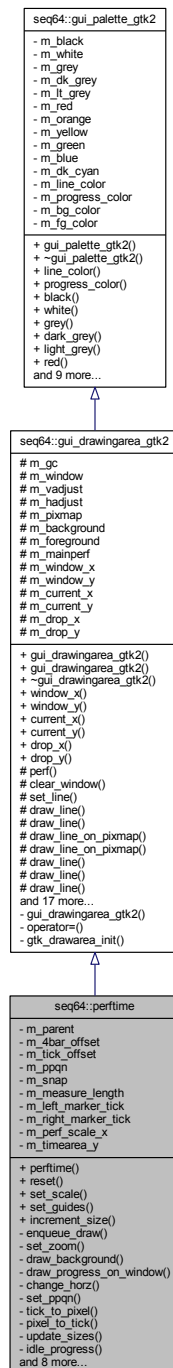
Even if the user specifies the fruity interaction, the Seq24 interaction is still needed to handle our new keystroke support for the `perfroll`. We need both objects to exist all the time, similar to the Fruity/Seq24 roles in the `seqroll` object.

Obsolete `AbstractPerfInput * m_interaction`

12.47 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](#), [perfedit](#) &parent, Gtk::Adjustment &hadjust, int ppqn=SEQ64_USE_DEFAULT_P↵PQN)

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 [enqueue_draw](#) ()

Wraps `queue_draw()` and forwards the call to the parent `perfdit`, so that it can forward it to any other `perfdit` that exists.

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

Implements the horizontal zoom feature.

- void [draw_background](#) ()

Separated out the drawing done in `on_expose_event()`, so that it can be redone when the zoom changes.

- 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](#) (midipulse tick)

Common calculation to convert a pulse/tick value to a perftime x value.

- [midipulse_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 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 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.

- bool [key_press_event](#) (GdkEventKey *ev)

This callback function handles a key-press event.

Private Attributes

- [perfedit](#) & [m_parent](#)
Provides a link to the perfedit that created this object.
- `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

12.47.1 Constructor & Destructor Documentation

12.47.1.1 `seq64::perftime::perftime (perform & p, perfedit & parent, Gtk::Adjustment & hadjust, 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.

12.47.2 Member Function Documentation

12.47.2.1 `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() .
Generated by Doxygen	

12.47.2.2 void seq64::perftime::enqueue_draw () [private]

The parent perfedit will call perftime::queue_draw() on behalf of this object, and it will pass a [perftime::enqueue_draw\(\)](#) to the peer perfedit's perftime, if the peer exists.

12.47.2.3 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);
```

12.47.2.4 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.

12.47.2.5 bool seq64::perftime::key_press_event (GdkEventKey * ev) [private]

Can't get the keystroke events to be seen by perftroll or perftime here using the normal callback function for keystrokes, and not sure why. The perfedit object can call this function, and that call works, so the perfedit class, which does get keystrokes, calls this function to do the work.

This function uses the "l" key to activate the movement of the "L" marker with the arrow keys, by the interval of on_snap value for each press. It also uses the "r" key to activate the movement of the "R" marker, and the "x" to deactivate either movement move.

Be aware that there is no visual feedback, as yet, that one is in the movement mode.

Also be aware the changing the name of this function from "key_press_event()" to "on_key_press_event()" will disrupt the process, causing keystrokes to not get here. Too tricky.

12.47.3 Field Documentation

12.47.3.1 perfedit& seq64::perftime::m_parent [private]

We want to support two perfedit windows, but the children of perfedit will have to communicate changes requiring a redraw through the parent.

12.47.3.2 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. Supports perftime's keystroke processing.

12.47.3.3 `int seq64::perftime::m_left_marker_tick [private]`

Otherwise it is -1. Supports perftime's keystroke processing.

12.47.3.4 `int seq64::perftime::m_right_marker_tick [private]`

Otherwise it is -1. Supports perftime's keystroke processing.

12.48 `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 auto_option_save () const`
Accessor m_auto_option_save
- `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 [with_jack](#) () const
- **Accessor** *m_with_jack_transport m_with_jack_master, and m_with_jack_master_cond, to save client code some trouble.*
- bool [jack_start_mode](#) () const
- **Accessor** *m_jack_start_mode,*
- bool [manual_alsa_ports](#) () const
- **Accessor** *m_manual_alsa_ports*
- bool [reveal_alsa_ports](#) () const
- **Accessor** *m_reveal_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.*

12.48.1 Member Function Documentation

12.48.1.1 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.

12.49 seq64::rect Class Reference

A small helper class representing a rectangle.

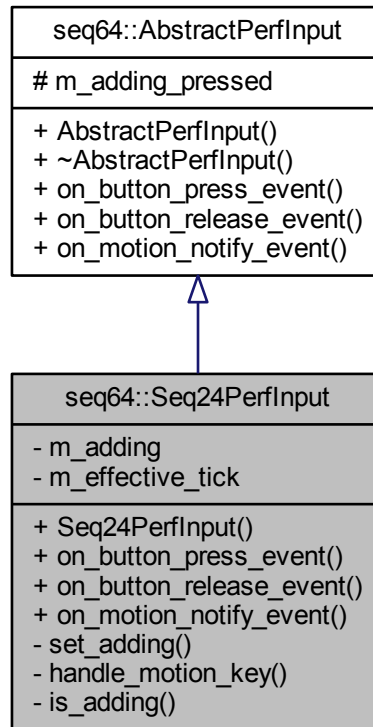
12.50 seq64::gui_drawingarea_gtk2::rect Struct Reference

A small helper structure representing a rectangle.

12.51 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.
- [midipulse m_effective_tick](#)
The current tick for the current segment?

12.51.1 Member Function Documentation

12.51.1.1 `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](#).

12.51.1.2 `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](#).

12.51.1.3 `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](#).

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

What does it mean?

12.51.1.5 `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.

12.52 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.

12.52.1 Member Function Documentation

12.52.1.1 void seq64::Seq24SeqEventInput::set_adding (bool adding, sequevent & seqev)

Modifies m_adding as well.

12.52.1.2 bool seq64::Seq24SeqEventInput::on_button_press_event (GdkEventButton * a_ev, sequevent & seqev)

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. seqev.m_seq.unselect(); ???????

12.52.1.3 `bool seq64::Seq24SeqEventInput::on_button_release_event (GdkEventButton * a_ev, sequevent & seqev)`

Returns

Returns true if a likely modification was made. This function used to return true all the time.

12.52.1.4 `bool seq64::Seq24SeqEventInput::on_motion_notify_event (GdkEventMotion * a_ev, sequevent & seqev)`

Returns

Returns true if a likely modification was made. This function used to return true all the time.

12.53 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.

12.53.1 Member Function Documentation

12.53.1.1 `void seq64::Seq24SeqRollInput::set_adding (bool adding, seqroll & sroll)`

What calls this? It is actually a right click.

12.53.1.2 `bool seq64::Seq24SeqRollInput::on_button_press_event (GdkEventButton * ev, seqroll & sroll)`

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.

12.53.1.3 `bool seq64::Seq24SeqRollInput::on_button_release_event (GdkEventButton * ev, seqroll & sroll)`

This function now uses the `needs_update` flag to determine if the perform object should modify().

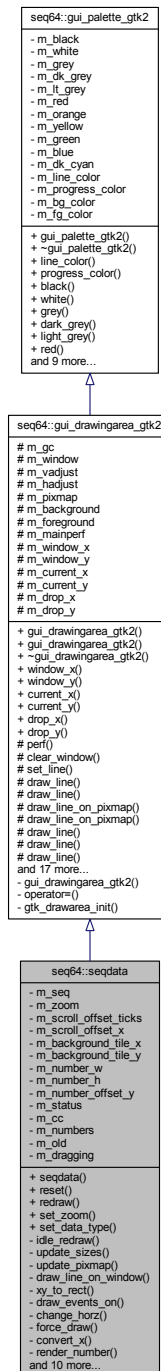
Returns

Returns the value of `needs_update`. It used to return only true.

12.54 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\(\)](#).*

Private Member Functions

- int [idle_redraw](#) ()
- Draws events on this object's built-in window and pixmap.*

 - 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, midipulse &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

12.54.1 Constructor & Destructor Documentation

12.54.1.1 `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.

12.54.2 Member Function Documentation

12.54.2.1 `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.

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

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

12.54.2.3 `void seq64::seqdata::set_data_type (midibyte status, midibyte control)`

Perhaps we should check that at least one of the parameters causes a change.

12.54.2.4 `int seq64::seqdata::idle_redraw () [private]`

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

12.54.2.5 void seq64::seqdata::update_sizes () [private]

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

12.54.2.6 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.

12.54.2.7 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).

12.54.2.8 bool seq64::seqdata::on_expose_event (GdkEventExpose * a_e) [private]

Returns

Always returns true.

12.54.2.9 bool seq64::seqdata::on_button_press_event (GdkEventButton * ev) [private]

Returns

Always returns true.

12.54.2.10 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.

12.54.2.11 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.

12.54.2.12 `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.

If there is a note (seqroll pane) or event (sequevent pane) selected, and mouse hovers over the data area (seqdata pane), then this scrolling action will increase or decrease the value of the data item, which lengthens or shortens the line drawn.

Todo DOCUMENT the seqdata scrolling behavior in the documentation projects.

Returns

Always returns true.

12.54.3 Field Documentation

12.54.3.1 `int seq64::seqdata::m_number_w [private]`

By "adjusted", well this is just a minor tweak for appearances.

12.54.3.2 `int seq64::seqdata::m_number_h [private]`

Basically, the character height times 3. By "adjusted", well this is just a minor tweak for appearances.

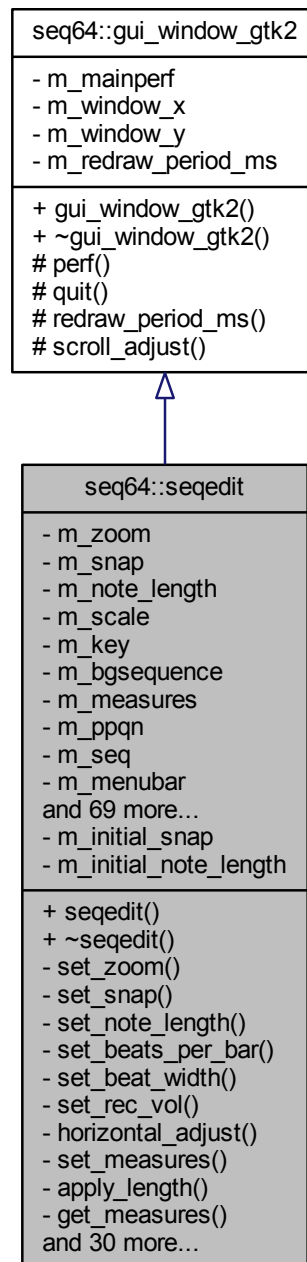
12.54.3.3 `int seq64::seqdata::m_number_offset_y [private]`

This value was hardwired as 8, for a character height of 10.

12.55 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 [horizontal_adjust](#) (double step)
This function provides optimization for the [on_scroll_event\(\)](#) function.
- 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 tells 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 [redraw_period_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: 1 2 3 4, and 1, 2, 4, 8, 16.
- int [m_snap](#)

Used in setting the snap-to value in pulses, off = 1.
- int [m_note_length](#)

The default length of a note to be inserted by a right-left-click operation.
- int [m_scale](#)

Settings for the music scale, key, and background sequence.
- long [m_measures](#)

Provides the length of the sequence in measures.
- int [m_ppqn](#)

Holds a copy of the current PPQN for the sequence (and the entire MIDI file).
- [sequence](#) & [m_seq](#)

Holds a reference to the sequence that this window represents.

- Gtk::MenuBar * [m_menubar](#)
A number of user-interface objects.
- 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.
- int [m_pos](#)
Basically the sequence number.
- Gtk::Adjustment * [m_vadjust](#)
Scrollbar and adjustment objects for horizontal and vertical panning.
- [seqkeys](#) * [m_seqkeys_wid](#)
Handles the piano-keys part of the user-interface.
- [seqtime](#) * [m_seqtime_wid](#)
Handles the time-line (bar or measures) part of the user-interface.
- [seqdata](#) * [m_seqdata_wid](#)
Handles the event-data part of the user-interface.
- [sequevent](#) * [m_sequevent_wid](#)
Handles the small event part of the user-interface, where events can be moved and added.
- [seqroll](#) * [m_seqroll_wid](#)
Handles the piano-roll part of the user-interface.
- Gtk::Table * [m_table](#)
More user-interface elements.
- 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

12.55.1 Detailed Description

- perform
- seqroll
- seqkeys
- seqdata
- seqtime
- sequevent
- sequence

This class has a metric ton of user-interface objects and other members.

12.55.2 Constructor & Destructor Documentation

12.55.2.1 `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.

Horizontal Gtk::Adjustment constructor: The initial value was 0 on a range from 0 to 1, with step and page increments of 1, and a page_size of 1. We can fix these values here, or create an `h_adjustment()` function similar to [eventedit::v_adjustment\(\)](#), which first gets called in `on_realize()`.

12.55.3 Member Function Documentation

12.55.3.1 `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.

12.55.3.2 `void seq64::seqedit::set_snap (int snap) [private]`

It is passed to the seqroll, sequevent, and sequence objects, as well.

12.55.3.3 `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
-------------------	---

12.55.3.4 `void seq64::seqedit::horizontal_adjust (double step) [inline], [private]`

A duplicate of the one in seqroll.

Parameters

<i>step</i>	Provides the step value to use for adjusting the horizontal scrollbar. See gui_drawingarea_gtk2::scroll_adjust() for more information.
-------------	--

12.55.3.5 `void seq64::seqedit::set_measures (int lim) [private]`

Parameters

<i>lim</i>	Provides the sequence length, in measures.
------------	--

12.55.3.6 `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 sequevent objects are reset().

12.55.3.7 `long seq64::seqedit::get_measures () [private]`

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

12.55.3.8 `void seq64::seqedit::set_midi_channel (int midichannel) [private]`

Should this change set the is-modified flag?

12.55.3.9 `void seq64::seqedit::set_midi_bus (int bus) [private]`

Should this change set the is-modified flag?

12.55.3.10 `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()`.

12.55.3.11 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()`.

12.55.3.12 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.

12.55.3.13 void seq64::seqedit::name_change_callback () [private]

That name is the name the user has given to the sequence being edited.

12.55.3.14 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!

12.55.3.15 void seq64::seqedit::fill_top_bar () [private]

Note that, if a non-default title for the sequence is in force, then we immediately force the focus to the seqroll "widget", so that the space bar can be used to control playback, instead of immediately erasing the name of the sequence.

12.55.3.16 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".

Note that many of the setups here could be loops through data structures. 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 one set the sequence length in bars. The second menu is the `m_menu_bpm`, or BPM, which here means "beats per measure" (not "beats per minute").

12.55.3.17 `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.

12.55.3.18 `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.

12.55.3.19 `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.

12.55.3.20 `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.

12.55.3.21 `bool seq64::seqedit::timeout () [private]`

Note the new call to `seqroll::follow_progress()`. This allows the `seqroll` to pop to the next frame of events to continue to show the moving progress bar. Does this need to be an option? It only affects patterns longer than a measure or two, whatever the width of the `seqroll` window is. This is a new feature that is not in `seq24`.

12.55.3.22 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).

12.55.3.23 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.

12.55.3.24 bool seq64::seqedit::on_scroll_event (GdkEventScroll * *ev*) [private]

This handles moving the scroll wheel on a mouse or do a two-fingered scrolling action on a touchpad. If no modifier key is pressed, this moves the view up or down on the "notes" coordinate, showing different piano keys. This behavior is implemented in [seqkeys::on_scroll_event\(\)](#), and is called into play by returning false here.

If the Ctrl key is pressed, then the scrolling action causes the view to zoom in or out. This behavior is implemented here.

If the Shift key is pressed, then the scrolling action moves the view horizontally on the time-line (measures-line) of the piano roll. This behavior is implemented here.

12.55.3.25 bool seq64::seqedit::on_key_press_event (GdkEventKey * *ev*) [private]

In particular, the Ctrl-W keypress. This keypress closes the sequence/pattern editor window by way of calling [on_delete_event\(\)](#). We should consider applying this convention to all the other windows.

12.55.4 Field Documentation

12.55.4.1 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.

12.55.4.2 `int seq64::seqedit::m_zoom` `[private]`

The value of zoom is the same as the number of pixels per tick on the piano roll.

12.55.4.3 `int seq64::seqedit::m_pos` `[private]`

We will eventually remove this member, as we can get it via [sequence::number\(\)](#) now.

12.55.4.4 `seqkeys* seq64::seqedit::m_seqkeys_wid` `[private]`

This item draws the piano-keys at the left of the seqedit window.

12.55.4.5 `seqtime* seq64::seqedit::m_seqtime_wid` `[private]`

This is the location where the measure numbers and the END marker are shown.

12.55.4.6 `seqdata* seq64::seqedit::m_seqdata_wid` `[private]`

This is the area at the bottom of the window that shows value lines for the selected kinds of events.

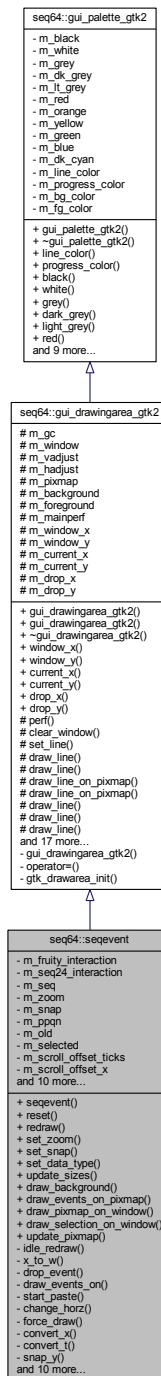
12.55.4.7 `Gtk::Table* seq64::seqedit::m_table` `[private]`

These items provide a number of buttons and text-entry fields, as well as their layout.

12.56 `seq64::sequevent` Class Reference

Implements the piano event drawing area.

Inheritance diagram for seq64::sequevent:



Public Member Functions

- [sequevent](#) ([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.

Private Member Functions

- int [idle_redraw](#) ()
Implements redraw while idling.
- 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](#) (midipulse 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, midipulse &tick)
Takes the screen x coordinate, multiplies it by the current zoom, and returns the tick value in the given parameter.
- void [convert_t](#) (midipulse ticks, int &x)
Converts the given tick value to an x coordinate, based on the zoom, and returns it via the second parameter.
- void [snap_y](#) (int &y)
This function performs a 'snap' on y.
- void [snap_x](#) (int &a_x)
This function performs a 'snap' on x.
- void [on_realize](#) ()
Implements the on-realize callback.
- bool [on_expose_event](#) (GdkEventExpose *a_ev)
Implements the on-expose event callback.

- bool [on_button_press_event](#) (GdkEventButton *a_ev)
Implements the on-button-press event callback.
- bool [on_button_release_event](#) (GdkEventButton *a_ev)
Implements the on-button-release event callback.
- bool [on_motion_notify_event](#) (GdkEventMotion *a_ev)
Implements the on-motion-notify event callback.
- bool [on_focus_in_event](#) (GdkEventFocus *)
Responds to a focus event by setting the HAS_FOCUS flag.
- bool [on_focus_out_event](#) (GdkEventFocus *)
Responds to a unfocus event by resetting the HAS_FOCUS flag.
- bool [on_key_press_event](#) (GdkEventKey *a_p0)
Implements the key-press event callback function.
- void [on_size_allocate](#) (Gtk::Allocation &)
Implements the on-size-allocate event callback.

Private Attributes

- FruitySeqEventInput [m_fruity_interaction](#)
Why should we need both at the same time? Just load the one that is specified in the configuration.
- int [m_zoom](#)
Zoom setting, means that one pixel == m_zoom ticks.
- bool [m_selecting](#)
Used when highlighting a bunch of events.
- midibyte [m_status](#)
Indicates what is the data window currently editing?

Additional Inherited Members

12.56.1 Member Function Documentation

12.56.1.1 void seq64::sequevent::redraw ()

Somewhat similar to [seqroll::redraw\(\)](#).

12.56.1.2 void seq64::sequevent::set_snap (int a_snap) [inline]

Simply sets the snap member.

12.56.1.3 void seq64::sequevent::set_data_type (midibyte status, midibyte control = 0)

Then redraws.

12.56.1.4 void seq64::sequevent::update_sizes ()

This ends up filling the background with dotted lines, etc.

12.56.1.5 `void seq64::segevent::draw_background ()`

It sets the foreground to white, draws the rectangle, in order to clear the pixmap.

12.56.1.6 `void seq64::segevent::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.
```

12.56.1.7 `int seq64::segevent::idle_redraw ()` [private]

Who calls this routine?

12.56.1.8 `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.

12.56.1.9 `void seq64::segevent::drop_event (midipulse 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`.

12.56.1.10 `void seq64::segevent::draw_events_on (Glib::RefPtr< Gdk::Drawable > draw)` [private]

Parameters

<i>draw</i>	The given drawable object.
-------------	----------------------------

12.56.1.11 `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.

12.56.1.12 `void seq64::segevent::change_horz ()` [private]

Very similar to [seqroll::change_horz\(\)](#).

12.56.1.13 `void seq64::segevent::convert_x (int x, midipulse & tick)` [inline], [private]

Why not just return it normally?

12.56.1.14 `void seq64::sequest::convert_t(midipulse ticks, int & x)` `[inline]`, `[private]`

Why not just return it normally?

12.56.1.15 `void seq64::sequest::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.

12.56.1.16 `void seq64::sequest::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.

12.56.1.17 `bool seq64::sequest::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.

12.56.1.18 `bool seq64::sequest::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.

12.56.1.19 `bool seq64::sequest::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.

12.56.1.20 `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. :-)

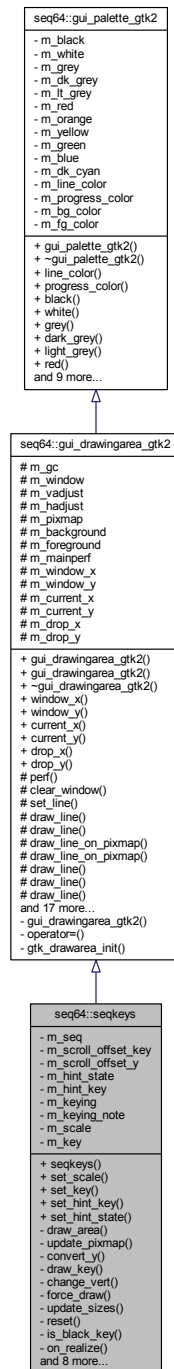
Returns

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.

12.57 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

12.57.1 Member Function Documentation

12.57.1.1 void seq64::seqkeys::set_hint_state (bool state)

Parameters

<i>state</i>	Provides the value for hinting, where true == on, false == off.
--------------	---

12.57.1.2 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 greyed when the mouse cursor is at the same vertical location on the piano as the key.

12.57.1.3 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.

12.57.1.4 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.

12.57.1.5 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.

12.57.1.6 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.

12.57.1.7 `bool seq64::seqkeys::on_enter_notify_event (GdkEventCrossing * p0)` [private]

I think this greys the current key.

12.57.1.8 `bool seq64::seqkeys::on_leave_notify_event (GdkEventCrossing * p0)` [private]

I think this un-greys the current key.

12.57.1.9 `bool seq64::seqkeys::on_scroll_event (GdkEventScroll * ev)` [private]

Note that there is no usage of the modifier keys (e.g. Shift or Ctrl). Compare this function to [seqedit::on_scroll_event\(\)](#).

Parameters

<i>ev</i>	Provides the direction of the scroll event.
-----------	---

Returns

Always returns true.

12.57.1.10 `void seq64::seqkeys::on_size_allocate (Gtk::Allocation & all)` [private]

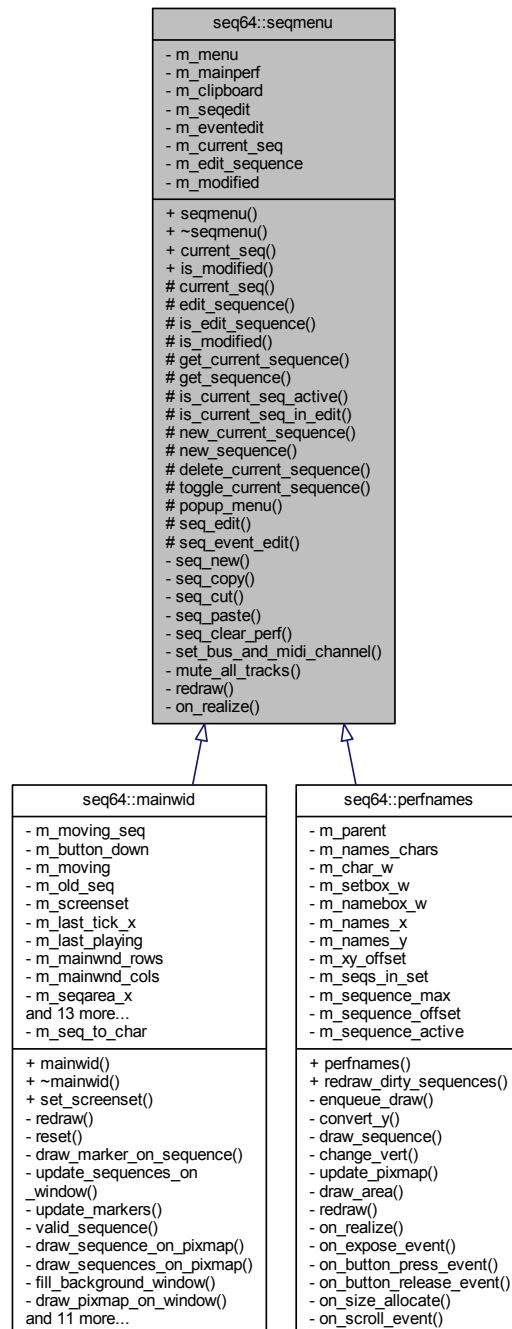
Parameters

<i>all</i>	Provies the allocation and its width and height.
------------	--

12.58 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_seq` () const

'Getter' function for member `m_current_seq` We're changing the name, so that "seq" indicates an integer by (an imperfect) convention.

- bool `is_modified` () const

'Getter' function for member `m_modified`

Protected Member Functions

- void `current_seq` (int seq)

'Setter' function for member `m_current_seq`

- void `edit_sequence` (int seqnum)

'Setter' function for member `m_edit_sequence` Pass in -1 to disable the edit-sequence number.

- bool `is_edit_sequence` (int seqnum) const

'Getter' function for member `m_edit_sequence` Tests the parameter against `m_edit_sequence`.

- void `is_modified` (bool flag)

'Setter' function for member `m_modified`

- `sequence * get_current_sequence` () const

'Getter' function for member `m_mainperf.get_sequence(current_seq())` This call is used many, many times, and well worth wrapping.

- `sequence * get_sequence` (int seqnum) const

Forwards the get-sequence call to the perform object.

- bool `is_current_seq_active` () const

Forwards the is-sequence-active check to the perform object.

- bool `is_current_seq_in_edit` () const

Forwards the is-sequence-in-edit check to the perform object.

- void `new_current_sequence` ()

Forwards the new-current-sequence call to the perform object.

- void `new_sequence` (int seqnum)

Forwards the new-sequence call to the perform object.

- void `delete_current_sequence` ()

Forwards the delete-sequence call to the perform object.

- void `toggle_current_sequence` ()

Forwards the sequence-playing-toggle call to the perform object.

- void `popup_menu` ()

This function sets up the File menu entries.

- void `seq_edit` ()

This menu callback launches the sequence-editor (pattern editor) window.

- void `seq_event_edit` ()

This menu callback launches the new event 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

- [perform](#) & [m_mainperf](#)

Provides a reference to the central object involved in managing a song and performance.

- [sequence](#) [m_clipboard](#)

Holds a copy of data concerning a sequence, which can then be pasted into another pattern slot.

- [seqedit](#) * [m_seqedit](#)

Points to the latest seqedit object, if created.

- [eventedit](#) * [m_eventedit](#)

Points to the latest eventedit object, if created.

- int [m_current_seq](#)

References the current sequence by sequence number.

- int [m_edit_sequence](#)

Hold the number of the currently-in-edit sequence.

- bool [m_modified](#)

Indicates if a sequence has been created.

12.58.1 Detailed Description

It is an abstract base class.

12.58.2 Constructor & Destructor Documentation

12.58.2.1 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.

12.58.2.2 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.

12.58.3 Member Function Documentation

12.58.3.1 `bool seq64::seqmenu::is_edit_sequence (int seqnum) const` `[inline], [protected]`

Returns true if that member is not -1, and the parameter matches it.

12.58.3.2 `void seq64::seqmenu::popup_menu ()` `[protected]`

It also sets up the pattern popup menu entries that are used in mainwid. Note that, for the selected sequence, the "Edit" and "Event Edit" menu entries are not included if a pattern editor or event editor is already running. The new event editor seems to create far-reaching problems that we do not yet understand, so it is now possible to disable it at build time. We have mitigated most of those problems by not allowing both a [seq_edit\(\)](#) and a [seq_event_edit\(\)](#) at the same time.

12.58.3.3 `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!

An oddity is that calling `show_all()` here does not work unless the `seqedit()` constructor makes its `show_all()` call.

12.58.3.4 `void seq64::seqmenu::seq_event_edit ()` `[protected]`

If it is already open for that sequence, this function just raises it.

Note that the `m_eventedit` member to which we save the new pointer is currently there just to avoid a compiler warning.

This menu entry is available only if the selected sequence is active. That is, if the sequence has already been created.

An oddity is that we need the `show_all()` call here in order to see the dialog. A situation different from that for `seqedit`! However, now it doesn't seem to be needed, and we have put it back into the `eventedit` constructor.

12.58.3.5 `void seq64::seqmenu::seq_new ()` `[private]`

For one thing, if `current_seq()` is either a -1 or is greater than the maximum allowed sequence number, [perform↵::is_active\(\)](#) will return false, and we have no idea whether the sequence is not active or the sequence number is just invalid. So we need to check the pointer we got before trying to use it.

12.58.3.6 `void seq64::seqmenu::seq_copy ()` `[private]`

We use a more appropriate function than operator `=()` here: [sequence::partial_assign\(\)](#).

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

12.58.3.7 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.

12.58.3.8 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.

12.58.3.9 void seq64::seqmenu::seq_clear_perf () [private]

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

12.58.4 Field Documentation

12.58.4.1 seqedit* seq64::seqmenu::m_seqedit [private]

Change Note Added by Chris on 2015-08-02 based on compiler warnings and a comment warning in the [seq_edit\(\)](#) function. We'll save the result of that function here, and will let valgrind tell us later if Gtkmm takes care of it.

12.58.4.2 bool seq64::seqmenu::m_modified [private]

Todo We need to make sure that the perform object is in control of the modification flag.

12.59 seq64::seqroll Class Reference

Implements the piano roll section of the pattern editor.

- 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 [update_and_draw](#) (int force=false)
Wraps up some common code.
- 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_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 selection on the main window.
- void [draw_progress_on_window](#) ()
Draw a progress line on the window.
- void [reset](#) ()
This function basically resets the whole widget as if it were realized again.
- void [redraw](#) ()
Redraws unless m_ignore_redraw is true.
- void [redraw_events](#) ()
Redraws events unless m_ignore_redraw is true.
- void [start_paste](#) ()
Starts a paste operation.
- void [follow_progress](#) ()
Checks the position of the tick, and, if it is in a different piano-roll "page" than the last page, moves the page to the next page.

Private Member Functions

- void [horizontal_adjust](#) (double step)
This function provides optimization for the [on_scroll_event\(\)](#) function.
- void [snap_y](#) (int &y)
Snaps the y value to the piano-key "height".
- void [snap_x](#) (int &x)
Performs a 'snap' operation on the x coordinate.
- void [convert_tn](#) (midipulse 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 [xy_to_rect](#) (int x1, int y1, int x2, int y2, int &x, int &y, int &w, int &h)
Converts rectangle corner coordinates to a starting coordinate, plus a width and height.
- void [convert_tn_box_to_rect](#) (midipulse tick_s, midipulse 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.
- int [idle_redraw](#) ()
Draw the events on the main window and on the pixmap.
- 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

- [Gtk::Adjustment](#) & [m_horizontal_adjust](#)
We need direct access to the horizontal scrollbar if we want to be able to make it follow the progress bar.
- [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_pos](#)
A position value.
- [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.
- [int](#) [m_scroll_page](#)
Provides the current scroll page in which the progress bar resides.

Friends

- [class FruitySeqRollInput](#)
These friend implement interaction-specific behavior, although only the Seq24 interactions support keyboard processing.

Additional Inherited Members

12.59.1 Constructor & Destructor Documentation

- 12.59.1.1 [seq64::seqroll::seqroll](#) ([perform](#) & *p*, [sequence](#) & *seq*, [int](#) *zoom*, [int](#) *snap*, [seqkeys](#) & *seqkeys_wid*, [int](#) *pos*, [Gtk::Adjustment](#) & *hadjust*, [Gtk::Adjustment](#) & *vadjust*, [int](#) *ppqn* = [SEQ64_USE_DEFAULT_PPQN](#))

Parameters

<i>p</i>	The performance object that helps control this piano roll. Note that we can get the perform object from the sequence, and save a parameter. Low priority change.
<i>seq</i>	The sequence object represented by this piano roll.
<i>zoom</i>	The initial zoom of this piano roll.
<i>snap</i>	The initial grid snap of this piano roll.
<i>seqkeys_wid</i>	A reference to the piano keys window that is shown to the left of this piano roll.
<i>pos</i>	A position parameter. See the description of seqroll::m_pos . This is actually the sequence number, and is currently unused. However, we're sure we can find a use for it sometime.
<i>hadjust</i>	Represents the horizontal scrollbar of this window. It is actually created by the "parent" seqedit object.
<i>vadjust</i>	Represents the vertical scrollbar of this window. It is actually created by the "parent" seqedit object.
<i>ppqn</i>	The initial value of the PPQN for this sequence. Useful in scale calculations.

12.59.1.2 seq64::seqroll::~~seqroll ()

The only thing to delete here is the clipboard.

12.59.2 Member Function Documentation

12.59.2.1 void seq64::seqroll::set_zoom (int *zoom*)

Parameters

<i>zoom</i>	The desired zoom value.
-------------	-------------------------

12.59.2.2 void seq64::seqroll::set_key (int *key*)

Parameters

<i>key</i>	The desired key value.
------------	------------------------

12.59.2.3 void seq64::seqroll::set_scale (int *scale*)

Parameters

<i>scale</i>	The desired scale value.
--------------	--------------------------

12.59.2.4 void seq64::seqroll::set_data_type (midibyte *status*, midibyte *control*)

Unlike the same function in sequevent, this version does not redraw.

12.59.2.5 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.
<i>seq</i>	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.

12.59.2.6 void seq64::seqroll::update_sizes ()

It brings the scrollbar back to the beginning, resets the upper limit to the number of ticks in the sequence, sets the page-size based on the window size and the zoom factor.

The horizontal step increment is 1 semiquaver (1/16) note per zoom level. The horizontal page increment is currently always one bar. We may want to make that larger for scrolling after the progress bar.

The maximum value set for the scrollbar brings it to the last "page" of the piano roll.

The vertical size are also adjusted. More on the story later.

12.59.2.7 void seq64::seqroll::update_background ()

The first thing done is to clear the background, painting it white.

12.59.2.8 void seq64::seqroll::draw_events_on_pixmap ()

Just calls [draw_events_on\(\)](#).

12.59.2.9 void seq64::seqroll::draw_progress_on_window ()

This is done by first blanking out the line with the background, which contains white space and grey lines, using the [draw_drawable](#) function. Remember that we wrap the [draw_drawable\(\)](#) function so its parameters are xsrc, ysrc, xdest, ydest, width, and height.

Note that the progress-bar position is based on the [sequence::get_last_tick\(\)](#) value, the current zoom, and the current scroll-offset x value.

12.59.2.10 void seq64::seqroll::reset ()

It's almost identical to the [change_horz\(\)](#) function, just calling [update_sizes\(\)](#) before [update_and_draw\(\)](#).

12.59.2.11 void seq64::seqroll::redraw ()

Somewhat similar to [sequevent::redraw\(\)](#).

12.59.2.12 void seq64::seqroll::follow_progress ()

We don't want to do any of this if the length of the sequence fits in the window, but for now it doesn't hurt; the progress bar just never meets the criterion for moving to the next page.

- Todo**
- If playback is disabled (such as by a trigger), then do not update the page;
 - When it comes back, make sure we're on the correct page;
 - When it stops, put the window back to the beginning, even if the beginning is not defined as "0".

12.59.2.13 void seq64::seqroll::horizontal_adjust (double step) [inline], [private]

A duplicate of the one in [seqedit](#).

Parameters

<i>step</i>	Provides the step value to use for adjusting the horizontal scrollbar. See gui_drawingarea_gtk2::scroll_adjust() for more information.
-------------	--

12.59.2.14 `void seq64::seqroll::snap_y (int & y) [inline],[private]`

Parameters

<i>y</i>	The y-value to be snapped.
----------	----------------------------

12.59.2.15 `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.

12.59.2.16 `void seq64::seqroll::convert_tn (midipulse ticks, int note, int & x, int & y) [private]`

This function is the "inverse" of `convert_xy()`.

Parameters

<i>tick</i>	Provides the horizontal value in MIDI pulses.
<i>note</i>	Provides the vertical value, a note value.
<i>x</i>	Provides the destination x value of the coordinate.
<i>y</i>	Provides the destination y value of the coordinate.

12.59.2.17 `void seq64::seqroll::xy_to_rect (int x1, int y1, int x2, int y2, int & x, int & y, int & w, int & h) [private]`

This function checks the mins / maxes, and then fills in the x, y, width, and height values.

12.59.2.18 `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.

12.59.2.19 `void seq64::seqroll::change_horz () [private]`

Roughly similar to [sequevent::change_horz\(\)](#).

12.59.2.20 `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.

12.59.2.21 `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. The seqedit class handles that fun stuff.

Note that this function seems to duplicate the functionality of [seqkeys::on_scroll_event\(\)](#). Do we really need both? Which one do we need?

12.59.3 Field Documentation

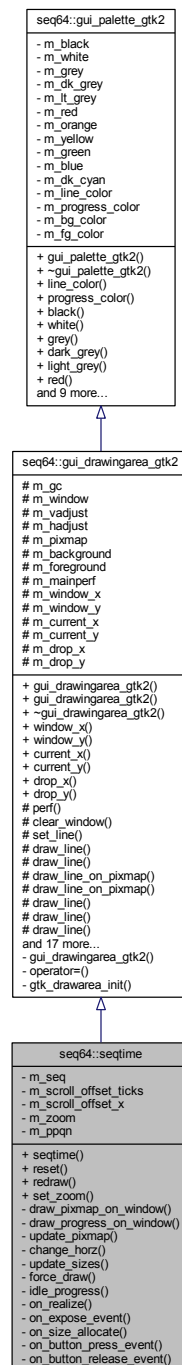
12.59.3.1 `int seq64::seqroll::m_pos [private]`

Need to clarify what exactly this member is used for.

12.60 seq64::seqtime Class Reference

This class implements the piano time, whatever that is.

Inheritance diagram for seq64::seqtime:



Public Member Functions

- **seqtime** (sequence &seq, perform &p, int zoom, Gtk::Adjustment &hadjust, int ppqn=SEQ64_USE_DEFAULT_PPQN)

Principal constructor.

- void **reset** ()

Sets the scroll offset tick and x values, updates the sizes and the pixmap, and resets the window.

- void [redraw](#) ()
Very similar to the [reset\(\)](#) function, except it doesn't update the sizes.
- void [set_zoom](#) (int zoom)
Sets the zoom to the given value and resets the window.

Private Member Functions

- void [draw_pixmap_on_window](#) ()
Draws the pixmap on the window.
- void [update_pixmap](#) ()
Updates the pixmap.
- void [change_horz](#) ()
Changes the scrolling horizontal offset, updates the pixmap, and forces a redraw.
- void [update_sizes](#) ()
Updates the pixmap to a new size and queues up a draw operation.
- void [force_draw](#) ()
Same as [draw_pixmap_on_window\(\)](#).
- bool [idle_progress](#) ()
Simply returns true.
- void [on_realize](#) ()
Called when the window is drawn.
- bool [on_expose_event](#) (GdkEventExpose *a_ev)
Implements the on-expose event handler.
- void [on_size_allocate](#) (Gtk::Allocation &)
Implements the on-size-allocate event handler.
- 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

12.60.1 Constructor & Destructor Documentation

- 12.60.1.1 [seq64::seqtime::seqtime](#) ([sequence](#) & seq, [perform](#) & p, int zoom, [Gtk::Adjustment](#) & hadjust, int ppqn = SEQ64_USE_DEFAULT_PPQN)

In the constructor you can only allocate colors; [get_window\(\)](#) returns 0 because the window is not yet realized>

12.60.2 Member Function Documentation

12.60.2.1 void seq64::seqtime::update_pixmap () [private]

When the zoom is at 32, there is a thick bar for every measure, and a measure number and major time division every 4 measures.at the default PPQN of 192.

Let me know if you figure out this legacy chart from the original seq24 code:

zoom	32	16	8	4	1
m1					
m_ppqn					
*					
1	128				
2	64				
4	32	16	8		
8	16m	8	4	2	1
16	8m	4	2	1	1
32	4m	2	1	1	1
64	2m	1	1	1	1
128	1m	1	1	1	1

A major line is a line that has a measure number in the timeline. The number of measures in a major line is 1 for zooms from 1:1 to 1:8; 2 for zoom 1:16; 4 for zoom 1:32; 8 for zoom 1:64 (new); and 16 for zoom 1:128. Zooms 1:64 and 1:128 look good only for high PPQN values.

12.60.2.2 void seq64::seqtime::on_realize () [private]

Call the base-class version of this function first. Then addition resources are allocated.

12.60.2.3 bool seq64::seqtime::on_button_press_event (GdkEventButton *) [inline],[private]

Simply returns false.

12.60.2.4 bool seq64::seqtime::on_button_release_event (GdkEventButton *) [inline],[private]

Simply returns false.

12.61 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

Public Member Functions

- [sequence](#) (int ppqn=SEQ64_USE_DEFAULT_PPQN)
Principal constructor.
- [~sequence](#) ()
A rote destructor.
- void [partial_assign](#) (const [sequence](#) &rhs)
A cut-down version of principal assignment operator.
- [event_list](#) & [events](#) ()
'Getter' function for member m_events
- const [event_list](#) & [events](#) () const
'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 event-list into the undo-list.
- void [pop_undo](#) ()
If there are items on the undo list, this function pushes the event-list into the redo-list, puts the top of the undo-list into the event-list, 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 event-list into the undo-list, puts the top of the redo-list into the event-list, 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.
- int [get_ppqn](#) () const
'Getter' function for member m_ppqn Provided as a convenience for the [editable_events](#) class.
- void [set_beats_per_bar](#) (int beatspermeasure)
'Setter' function for member m_time_beats_per_measure
- int [get_beats_per_bar](#) () const
'Getter' function for member m_time_beats_per_measure
- void [set_beat_width](#) (int beatwidth)
'Setter' function for member m_time_beat_width
- int [get_beat_width](#) () const
'Getter' function for member m_time_beat_width

- void `clocks_per_metronome` (int cpm)
'Setter' function for member m_clocks_per_metronome
- int `clocks_per_metronome` () const
'Getter' function for member m_clocks_per_metronome
- void `set_32nds_per_quarter` (int tpq)
'Setter' function for member m_32nds_per_quarter
- int `get_32nds_per_quarter` () const
'Getter' function for member m_32nds_per_quarter
- void `us_per_quarter_note` (int upqn)
'Setter' function for member m_us_per_quarter_note
- int `us_per_quarter_note` () const
'Getter' function for member m_us_per_quarter_note
- void `set_rec_vol` (int 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` (midipulse len, bool adjust_triggers=true)
Sets the length (m_length) and adjusts triggers for it, if desired.
- midipulse `get_length` () const
'Getter' function for member m_length
- midipulse `get_last_tick` ()
Returns the last tick played, and is used by the editor's idle function.
- void `set_last_tick` (midipulse tick)
'Setter' function for member m_last_tick This function used to be called "set_orig_tick()", now renamed to match up with get_last_tick().
- midipulse `mod_last_tick` ()
Some MIDI file errors and other things can lead to an m_length of 0, which causes arithmetic errors when m_last_tick is modded against it.
- 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*
- `midipulse get_queued_tick` () const
- *'Getter' function for member m_queued_tick*
- bool `check_queued_tick` (`midipulse` 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*
- bool `is_smf_0` () const
- *Returns true if this sequence is an SMF 0 sequence.*
- void `set_midi_channel` (`midibyte` ch)
- *Sets the m_midi_channel number.*
- void `print` () const
- *Prints a list of the currently-held events.*
- void `print_triggers` () const
- *Prints a list of the currently-held triggers.*
- void `play` (`midipulse` tick, bool playback_mode)
- *The `play()` function dumps notes starting from the given tick, and it pre-buffers ahead.*
- bool `add_event` (const `event` &er)
- *Adds an event to the internal event list in a sorted manner.*
- void `add_trigger` (`midipulse` tick, `midipulse` len, `midipulse` offset=0, bool adjust_offset=true)
- *Adds a trigger.*
- void `split_trigger` (`midipulse` tick)
- *Splits a trigger.*

- void `grow_trigger` (`midipulse` tick_from, `midipulse` tick_to, `midipulse` len)
Grows a trigger.
- void `del_trigger` (`midipulse` tick)
Deletes a trigger, that brackets the given tick, from the trigger-list.
- bool `get_trigger_state` (`midipulse` tick)
Checks the list of triggers against the given tick.
- bool `select_trigger` (`midipulse` tick)
Checks the list of triggers against the given tick.
- bool `unselect_triggers` ()
Unselects all triggers.
- bool `intersect_triggers` (`midipulse` position, `midipulse` &start, `midipulse` &ender)
This function examines each trigger in the trigger list.
- bool `intersect_notes` (`midipulse` position, `midipulse` position_note, `midipulse` &start, `midipulse` &ender, int ¬e)
This function examines each note in the event list.
- bool `intersect_events` (`midipulse` posstart, `midipulse` posend, `midibyte` status, `midipulse` &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` (`midipulse` tick, bool adjust_offset, int which=2)
Moves selected triggers as per the given parameters.
- `midipulse` `selected_trigger_start` ()
Gets the last-selected trigger's start tick.
- `midipulse` `selected_trigger_end` ()
Gets the selected trigger's end tick.
- `midipulse` `get_max_trigger` ()
Get the ending value of the last trigger in the trigger-list.
- void `move_triggers` (`midipulse` start_tick, `midipulse` distance, bool direction)
Moves triggers in the trigger-list.
- void `copy_triggers` (`midipulse` start_tick, `midipulse` distance)
Copies triggers to...
- void `clear_triggers` ()
Clears the whole list of triggers.
- `midipulse` `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` (`midipulse` tick_s, int note_h, `midipulse` 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` (`midipulse` tick_s, `midipulse` 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 [cut_selected](#) (bool copyevents=true)
Cuts the selected events.
- void [paste_selected](#) (midipulse tick, int note)
Pastes the selected notes (and only note events) at the given tick and the given note value.
- void [get_selected_box](#) (midipulse &tick_s, int ¬e_h, midipulse &tick_f, int ¬e_l)
Returns the 'box' of the selected items.
- void [get_clipboard_box](#) (midipulse &tick_s, int ¬e_h, midipulse &tick_f, int ¬e_l)
Returns the 'box' of the clipboard items.
- void [move_selected_notes](#) (midipulse deltatick, int deltanote)
Removes and adds reads selected in position.
- void [add_note](#) (midipulse tick, midipulse len, int note, bool paint=false)
Adds a note of a given length and note value, at a given tick location.
- void [add_event](#) (midipulse 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](#) (midipulse tick_s, midipulse 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](#) (midipulse deltatick)
Moves note off event.
- void [stretch_selected](#) (midipulse 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 events in the event-list.
- 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 `pause` ()
EXPERIMENTAL.
- void `reset` (bool live_mode)
Provides a helper function simplify and speed up `perform::reset_sequences()`.
- 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` (midipulse *tick_s, midipulse *tick_f, int *note, bool *selected, int *velocity)
Each call to `seqdata()` fills the passed references with a events elements, and returns true.
- bool `get_minmax_note_events` (int &lowest, int &highest)
A new function provided so that we can find the minimum and maximum notes with only one (not two) traversal of the event list.
- bool `get_next_event` (midibyte status, midibyte cc, midipulse *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` (midipulse *tick_on, midipulse *tick_off, bool *selected, midipulse *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, midipulse 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`
- int `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.
- void `show_events` () const
A member function to dump a summary of events stored in the event-list of a sequence.
- void `copy_events` (const event_list &newevents)
Copies an external container of events into the current container, effectively replacing all of its events.

Private Member Functions

- void `set_parent` (`perform` *p)
'Setter' function for member m_parent Sets the "parent" of this sequence, so that it can get some extra information about the performance.
- void `put_event_on_bus` (`event` &ev)
Takes an event that this sequence is holding, and places it on the MIDI buss.
- void `set_trigger_offset` (`midipulse` trigger_offset)
Sets m_trigger_offset and wraps it to m_length.
- void `split_trigger` (`trigger` &trig, `midipulse` splittick)
Splits the trigger given by the parameter into two triggers.
- void `adjust_trigger_offsets_to_length` (`midipulse` 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 event-list.
- void `remove` (`event` &e)
A helper function, which does not lock/unlock, so it is unsafe to call without supplying an iterator from the event-list.
- void `remove_all` ()
Clears all events from the event container.

Private Attributes

- `perform` * `m_parent`
For pause support, we need a way for the sequence to find out if JACK transport is active.
- `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.
- bool `m_raise`
Used in seqmenu and seqedit.

- `std::string m_name`
Provides the name/title for the sequence.
- `midipulse 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.
- `midipulse m_length`
Holds the length of the sequence in pulses (ticks).
- `midipulse m_snap_tick`
The size of snap in units of pulses (ticks).
- `int m_time_beats_per_measure`
Provides the number of beats per bar used in this sequence.
- `int m_time_beat_width`
Provides with width of a beat.
- `int m_clocks_per_metronome`
Augments the beats/bar and beat-width with the additional values included in a Time Signature meta event.
- `int m_32nds_per_quarter`
Augments the beats/bar and beat-width with the additional values included in a Time Signature meta event.
- `int m_us_per_quarter_note`
Augments the beats/bar and beat-width with the additional values included in a Tempo meta event.
- `int m_rec_vol`
The volume to be used when recording.
- `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.
- `int 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.

12.61.1 Detailed Description

More members than you can shake a stick at.

12.61.2 Member Enumeration Documentation

12.61.2.1 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.

12.61.3 Member Function Documentation

12.61.3.1 void seq64::sequence::partial_assign (const sequence & rhs)

We're replacing that incomplete function (many members are not assigned) with the more accurately-named [partial_assign\(\)](#) function.

It did not assign them all, so we created this [partial_assign\(\)](#) function to do this work, and replaced operator =() with this function in client code.

Threadsafe

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

Threadsafe

12.61.3.3 void seq64::sequence::push_undo ()

Threadsafe

12.61.3.4 void seq64::sequence::pop_undo ()

Threadsafe

12.61.3.5 void seq64::sequence::pop_redo ()

Threadsafe

12.61.3.6 void seq64::sequence::push_trigger_undo ()

Threadsafe

12.61.3.7 void seq64::sequence::set_beats_per_bar (int beatspermeasure)

Threadsafe

Parameters

<i>beatspermeasure</i>	The new setting of the beats-per-bar value.
------------------------	---

12.61.3.8 void seq64::sequence::set_beat_width (int *beatwidth*)

Threadsafe

Parameters

<i>beatwidth</i>	The new setting of the beat width value.
------------------	--

12.61.3.9 int seq64::sequence::get_beat_width () const [inline]

Threadsafe

12.61.3.10 void seq64::sequence::set_rec_vol (int *recvol*)

Threadsafe

Parameters

<i>recvol</i>	The new setting of the recording volume setting.
---------------	--

12.61.3.11 const char* seq64::sequence::get_name () const [inline]

Deprecated

12.61.3.12 void seq64::sequence::set_length (midipulse *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

12.61.3.13 midipulse seq64::sequence::get_last_tick ()

If m_length is 0, this function returns m_last_tick - m_trigger_offset, to avoid an arithmetic exception. Should we return 0 instead?

Note that seqroll calls this function to help get the location of the progress bar. What does perfedit do?

12.61.3.14 void seq64::sequence::set_last_tick (midipulse tick)

Threadsafe

12.61.3.15 midipulse seq64::sequence::mod_last_tick () [inline]

This function replaces the "m_last_tick % m_length", returning m_last_tick if m_length is 0 or 1.

12.61.3.16 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.
----------	--

12.61.3.17 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

12.61.3.18 void seq64::sequence::off_queued ()

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

Threadsafe

12.61.3.19 void seq64::sequence::set_recording (bool r)

Threadsafe

12.61.3.20 void seq64::sequence::set_snap_tick (int st)

Threadsafe

12.61.3.21 void seq64::sequence::set_quantized_rec (bool qr)

Threadsafe

12.61.3.22 void seq64::sequence::set_thru (bool r)

Threadsafe

12.61.3.23 bool seq64::sequence::is_dirty_main ()

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

Threadsafe

12.61.3.24 bool seq64::sequence::is_dirty_edit ()

Threadsafe

12.61.3.25 bool seq64::sequence::is_dirty_perf ()

Threadsafe

12.61.3.26 bool seq64::sequence::is_dirty_names ()

Threadsafe

12.61.3.27 void seq64::sequence::set_dirty_mp ()

Not threadsafe

12.61.3.28 void seq64::sequence::set_dirty ()

Threadsafe

12.61.3.29 void seq64::sequence::set_midi_channel (midibyte ch)

Threadsafe

12.61.3.30 void seq64::sequence::print () const

Not threadsafe

12.61.3.31 `void seq64::sequence::print_triggers () const`

Not threadsafe

12.61.3.32 `void seq64::sequence::play (midipulse 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.

Note

With pause support, the progress bar for the pattern/sequence editor does what we want: pause with the pause button, and rewind with the stop button. Works with JACK, with issues, but we'd like to have the stop button do a rewind in JACK, too.

Parameters

<i>tick</i>	Provides the current end-tick value.
<i>playback_mode</i>	Provides how playback is managed. True indicates that it is performance/song-editor playback, controlled by the set of patterns and triggers set up in that editor, and saved with the song in seq24 format. False indicates that the playback is controlled by the main windows, in live mode.

Threadsafe

12.61.3.33 `bool seq64::sequence::add_event (const event & er)`

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.

Parameters

<i>er</i>	Provide a reference to the event to be added; the event is copied into the events container.
-----------	--

12.61.3.34 `void seq64::sequence::add_trigger (midipulse tick, midipulse len, midipulse offset = 0, bool fixoffset = true)`

A pass-through function that calls [triggers::add\(\)](#).

12.61.3.35 `void seq64::sequence::split_trigger (midipulse splittick)`

This is the public overload of `split_trigger`.

Threadsafe

12.61.3.36 `void seq64::sequence::grow_trigger (midipulse tickfrom, midipulse tickto, midipulse 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 <i>tickto</i> for the check.

Threadsafe

12.61.3.37 `void seq64::sequence::del_trigger (midipulse tick)`

Threadsafe

12.61.3.38 `bool seq64::sequence::get_trigger_state (midipulse 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.

12.61.3.39 `bool seq64::sequence::select_trigger (midipulse 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.

12.61.3.40 bool seq64::sequence::unselect_triggers ()**Returns**

Always returns false.

12.61.3.41 bool seq64::sequence::intersect_triggers (midipulse *position*, midipulse & *start*, midipulse & *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.

12.61.3.42 bool seq64::sequence::intersect_notes (midipulse *position*, midipulse *position_note*, midipulse & *start*, midipulse & *ender*, int & *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. Why is this an int value???

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.

12.61.3.43 `bool seq64::sequence::intersect_events (midipulse posstart, midipulse posend, midibyte status, midipulse & 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.

12.61.3.44 `void seq64::sequence::paste_trigger ()`

Why isn't this protected by a mutex? We will eventually enable this if anything bad happens, such as a deadlock, or corruption.

12.61.3.45 `bool seq64::sequence::move_selected_triggers_to (midipulse tick, bool adjustoffset, int which = 2)`

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

The `\a which` parameter has three possible values:

```
-# 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

Parameters

<i>tick</i>	The tick at which the trigger starts.
<i>adjustoffset</i>	Set to true if the offset is to be adjusted.
<i>which</i>	Selects which movement will be done, as discussed above.

Returns

Returns the value of [triggers::move_selected\(\)](#), which indicate that the movement could be made. Used in [Seq24Perflnput::handle_motion_key\(\)](#).

12.61.3.46 **midipulse** 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.

12.61.3.47 **midipulse** seq64::sequence::selected_trigger_end ()

Threadsafe

12.61.3.48 **midipulse** seq64::sequence::get_max_trigger ()

Threadsafe

12.61.3.49 **void** seq64::sequence::move_triggers (**midipulse** starttick, **midipulse** distance, **bool** direction)

Note the dependence on the m_length member being kept in sync with the parent's value of m_length.

Threadsafe

12.61.3.50 **void** seq64::sequence::copy_triggers (**midipulse** starttick, **midipulse** distance)

Threadsafe

12.61.3.51 **void** seq64::sequence::clear_triggers ()

Threadsafe

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

Threadsafe

12.61.3.53 **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.
------------	---

12.61.3.54 `int seq64::sequence::select_note_events (midipulse a_tick_s, int a_note_h, midipulse a_tick_f, int a_note_l, select_action_e a_action)`

Returns the number selected.

Threadsafe

12.61.3.55 `int seq64::sequence::select_events (midipulse tick_s, midipulse tick_f, midibyte status, midibyte cc, select_action_e action)`

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

Threadsafe

12.61.3.56 `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.

Parameters

<i>status</i>	Provides the status value to be selected.
<i>cc</i>	If the status is EVENT_CONTROL_CHANGE, then data byte 0 must match this value.
<i>inverse</i>	If true, invert the selection.

Returns

Always returns 0.

12.61.3.57 `int seq64::sequence::get_num_selected_notes () const`

Threadsafe

12.61.3.58 `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

12.61.3.59 `void seq64::sequence::select_all ()`

Threadsafe

12.61.3.60 void seq64::sequence::copy_selected ()

Threadsafe

12.61.3.61 void seq64::sequence::cut_selected (bool *copyevents* = true)

Threadsafe

12.61.3.62 void seq64::sequence::paste_selected (midipulse *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

12.61.3.63 void seq64::sequence::get_selected_box (midipulse & *tick_s*, int & *note_h*, midipulse & *tick_f*, int & *note_l*)

Note the common-code between this function and [get_clipboard_box\(\)](#).

Threadsafe

12.61.3.64 void seq64::sequence::get_clipboard_box (midipulse & *tick_s*, int & *note_h*, midipulse & *tick_f*, int & *note_l*)

Note the common-code between this function and [get_selected_box\(\)](#).

Threadsafe

12.61.3.65 void seq64::sequence::move_selected_notes (midipulse *delta_tick*, int *delta_note*)

Parameters

<i>delta_tick</i>	Provides the amount of time to move the selected notes.
<i>delta_note</i>	Provides the amount of pitch to move the selected notes.

12.61.3.66 void seq64::sequence::add_note (midipulse *tick*, midipulse *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

```
12.61.3.67 void seq64::sequence::add_event ( midipulse 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

```
12.61.3.68 void seq64::sequence::stream_event ( event & ev )
```

Threadsafe

```
12.61.3.69 bool seq64::sequence::change_event_data_range ( midipulse tick_s, midipulse 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</i> ↔ _s	Provides the starting tick value.
<i>tick</i> ↔ _f	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</i> ↔ _s	Provides the starting data value.
<i>data</i> ↔ _f	Provides the finishing data value.

Returns

Returns true if the data was changed.

12.61.3.70 `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

12.61.3.71 `void seq64::sequence::decrement_selected (midibyte astat, midibyte control)`

The supported statuses are:

- One-byte messages
 - EVENT_PROGRAM_CHANGE
 - EVENT_CHANNEL_PRESSURE
- Two-byte messages
 - EVENT_NOTE_ON
 - EVENT_NOTE_OFF
 - EVENT_AFTERTOUCH
 - EVENT_CONTROL_CHANGE
 - EVENT_PITCH_WHEEL

Threadsafe

12.61.3.72 `void seq64::sequence::grow_selected (midipulse delta_tick)`

If an event is not linked, this function now ignores the event's timestamp, rather than risk a segfault on a null pointer.

Threadsafe

Parameters

<i>delta_tick</i>	An offset for each linked event's timestamp.
-------------------	--

12.61.3.73 `void seq64::sequence::stretch_selected (midipulse 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

12.61.3.74 `void seq64::sequence::remove_marked ()`

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

Threadsafe

12.61.3.75 `void seq64::sequence::mark_selected ()`

Threadsafe

12.61.3.76 `void seq64::sequence::unpaint_all ()`

Threadsafe

12.61.3.77 `void seq64::sequence::unselect ()`

Threadsafe

12.61.3.78 `void seq64::sequence::verify_and_link ()`

Threadsafe

12.61.3.79 `void seq64::sequence::link_new ()`

Threadsafe

12.61.3.80 `void seq64::sequence::zero_markers () [inline]`

This function is used when the sequencer stops. This function currently sets `m_last_tick = 0`, but we would like to avoid that if doing a pause, rather than a stop, of playback. However, commenting out this setting doesn't have any effect that we can see with a quick look at the user-interface.

Parameters

<i>tick</i>	Provides the optional tick value to set as "0". It defaults to 0.
-------------	---

12.61.3.81 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

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

Threadsafe

12.61.3.83 void seq64::sequence::off_playing_notes ()

This function does not bother checking if m_masterbus is a null pointer.

Threadsafe

12.61.3.84 void seq64::sequence::pause ()

A pause version of [reset\(\)](#). The [reset\(\)](#) function is currently not called when pausing, but we still need the note-shutoff capability to prevent notes from lingering. Not that we do not call set_playing(false)... it disarms the sequence, which we do not want upon pausing.

12.61.3.85 void seq64::sequence::reset (bool *live_mode*)

Note that, in live mode, the user controls playback, while otherwise JACK or the performance/song editor controls playback. (We're still a bit confounded about these modes, alas.)

Parameters

<i>live_mode</i>	True if live mode is on. This means that JACK transport is not in control of playback.
------------------	--

12.61.3.86 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

12.61.3.87 void seq64::sequence::reset_draw_trigger_marker ()

Threadsafe

12.61.3.88 draw_type seq64::sequence::get_next_note_event (midipulse * *a_tick_s*, midipulse * *a_tick_f*, int * *a_note*, bool * *a_selected*, int * *a_velocity*)

When it has no more events, returns a false.

12.61.3.89 `bool seq64::sequence::get_minmax_note_events (int & lowest, int & highest)`

Todo For efficiency, we should calculate this only when the event set changes, and save the results and return them if good.

Threadsafe

Parameters

<i>lowest</i>	A reference parameter to return the note with the lowest value. if there are no notes, then it is set to SEQ64_MIDI_COUNT_MAX-1.
<i>highest</i>	A reference parameter to return the note with the highest value. if there are no notes, then it is set to -1.

Returns

If there are no notes in the list, then false is returned, and the results should be disregarded.

12.61.3.90 `bool seq64::sequence::get_next_event (midibyte status, midibyte cc, midipulse * 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.

12.61.3.91 `bool seq64::sequence::get_next_event (midibyte * a_status, midibyte * a_cc)`

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

12.61.3.92 `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 <code>std::list</code> 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.

12.61.3.93 `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.

12.61.3.94 void seq64::sequence::background_sequence (int *bs*) [inline]

Disabling the sequence number (setting it to SEQ64_SEQUENCE_LIMIT) is valid.

12.61.3.95 void seq64::sequence::copy_events (const event_list & *newevents*)

Compare this function to the [remove_all\(\)](#) function. Copying the container is a lot of work, but fairly fast, even with an std::multimap as the container.

Threadsafe Note that we had to consolidate the replacement of all the events in the container in order to prevent the "Save to Sequence" button in the eventedit object from causing the application to segfault. It would segfault when the mainwnd timer callback would fire, causing updates to the sequence's slot pixmap, which would then try to access deleted events. Part of the issue was that note links were dropped when copying the events, so now we call [verify_and_link\(\)](#) to hopefully reconstitute the links.

Parameters

<i>newevents</i>	Provides the container of MIDI events that will completely replace the current container. Normally this container is supplied by the event editor, via the eventslots class.
------------------	--

12.61.3.96 void seq64::sequence::set_parent (perform * *p*) [private]

Remember that m_parent is not at all owned by the sequence. We just don't want to do all the work necessary to make it a reference, at this time.

12.61.3.97 void seq64::sequence::put_event_on_bus (event & *ev*) [private]

This function does not bother checking if m_masterbus is a null pointer.

Parameters

<i>ev</i>	The event to put on the buss.
-----------	-------------------------------

Threadsafe

12.61.3.98 void seq64::sequence::set_trigger_offset (midipulse *trigger_offset*) [private]

If m_length is 0, then m_trigger_offset is simply set to the parameter.

Threadsafe

Parameters

<i>trigger_offset</i>	The full trigger offset to set.
-----------------------	---------------------------------

12.61.3.99 `void seq64::sequence::split_trigger (trigger & trig, midipulse 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.

12.61.3.100 `void seq64::sequence::adjust_trigger_offsets_to_length (midipulse newlength)` `[private]`

Threadsafe

Might can get rid of this function?

12.61.3.101 `void seq64::sequence::remove (event_list::iterator i)` `[private]`

(And we no longer bother checking the pointer. If it is now, all hope is lost.) If it's a note off, and that note is currently playing, then send a note off. Do we need a `flush()` call as well?

Not threadsafe

12.61.3.102 `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()`!

12.61.3.103 `void seq64::sequence::remove_all ()` `[private]`

Unsets the modified flag. Also see the new `copy_events()` function.

12.61.4 Field Documentation

12.61.4.1 `perform* seq64::sequence::m_parent` `[private]`

We can use the `rc_settings` flag(s), but JACK could be disconnected. We could use a reference here, but, to avoid modifying the midifile class as well, we use a pointer. It is set in `perform::add_sequence()`.

12.61.4.2 midibyte seq64::sequence::m_midi_channel [private]

However, if this value is EVENT_NULL_CHANNEL (0xFF), then this sequence is an SMF 0 track, and has no single channel.

12.61.4.3 int seq64::sequence::m_playing_notes[SEQ64_MIDI_NOTES_MAX] [private]

It is used when muting, to shut off the notes that are playing.

12.61.4.4 bool seq64::sequence::m_raise [private]

It allows a sequence editor window to pop up if not already raised, in [seqedit::timeout\(\)](#).

12.61.4.5 int seq64::sequence::m_seq_number [private]

This number is set in the [perform::install_sequence\(\)](#) function.

12.61.4.6 midipulse seq64::sequence::m_length [private]

This value should be a power of two when used as a bar unit.

12.61.4.7 midipulse seq64::sequence::m_snap_tick [private]

It starts out as the value m_ppqn / 4.

12.61.4.8 int seq64::sequence::m_time_beats_per_measure [private]

Defaults to 4. Used by the sequence editor to mark things in correct time on the user-interface.

12.61.4.9 int seq64::sequence::m_time_beat_width [private]

Defaults to 4, which means the beat is a quarter note. A value of 8 would mean it is an eighth note. Used by the sequence editor to mark things in correct time on the user-interface.

12.61.4.10 int seq64::sequence::m_clocks_per_metronome [private]

This value provides the number of MIDI clocks between metronome clicks. The default value of this item is 24. It can also be read from some SMF 1 files, such as our hymne.mid example.

12.61.4.11 int seq64::sequence::m_32nds_per_quarter [private]

This value provides the number of notated 32nd notes in a MIDI quarter note (24 MIDI clocks). The usual (and default) value of this parameter is 8; some sequencers allow this to be changed.

12.61.4.12 `int seq64::sequence::m_us_per_quarter_note` `[private]`

This value can be extracted from the beats-per-minute value ([mastermidibus::m_beats_per_minute](#)), but here we set it to 0 by default, indicating that we don't want to write it. Otherwise, it can be read from a MIDI file, and saved here to be restored later.

12.61.4.13 `midibyte seq64::sequence::m_musical_key` `[private]`

If the value is SEQ64_KEY_OF_C, then there is no musical key to be set.

12.61.4.14 `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.

12.61.4.15 `int seq64::sequence::m_background_sequence` `[private]`

If the value is greater than `max_sequence()`, then there is no background sequence to be set.

12.61.4.16 `mutex seq64::sequence::m_mutex` `[mutable], [private]`

Made mutable for use in certain locked getter functions.

12.62 `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.
- `midipulse tick_start` () const
'Getter' function for member `m_tick_start`
- `void tick_start` ([midipulse](#) s)
'Setter' function for member `m_tick_start`
- `void increment_tick_start` ([midipulse](#) s)
'Setter' function for member `m_tick_start`
- `void decrement_tick_start` ([midipulse](#) s)
'Setter' function for member `m_tick_start`
- `midipulse tick_end` () const
'Getter' function for member `m_tick_end`
- `void tick_end` ([midipulse](#) e)
'Setter' function for member `m_tick_end`

- void [increment_tick_end](#) (midipulse s)
'Setter' function for member m_tick_end
- void [decrement_tick_end](#) (midipulse s)
'Setter' function for member m_tick_end
- [midipulse offset](#) () const
'Getter' function for member m_offset
- void [offset](#) (midipulse o)
'Setter' function for member m_offset
- void [increment_offset](#) (midipulse s)
'Setter' function for member m_offset
- void [decrement_offset](#) (midipulse 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

- [midipulse m_tick_start](#)
Provides the starting tick for this trigger.
- [midipulse m_tick_end](#)
Provides the ending tick for this trigger.
- [midipulse m_offset](#)
Provides the offset for this trigger.
- bool [m_selected](#)
Indicates that the trigger is part of a selection.

12.62.1 Detailed Description

This class is used in playback, and is contained in the triggers class.

12.62.2 Member Function Documentation

12.62.2.1 bool seq64::trigger::operator< (const trigger & rhs) [inline]

Parameters

<i>rhs</i>	The "right-hand side" of the less-than operation.
------------	---

Returns

Returns true if m_tick_start is less than rhs's.

12.63 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) const`
Prints a list of the currently-held triggers.
- `bool play (midipulse &starttick, midipulse &endtick)`
If playback-mode (song mode) is in force, that is, if using in-triggers and on/off triggers, this function handles that kind of playback.
- `void add (midipulse tick, midipulse len, midipulse offset=0, bool adjustoffset=true)`
Adds a trigger.
- `void adjust_offsets_to_length (midipulse newlen)`
Adjusts trigger offsets to the length of ???, for all triggers, and undo triggers.
- `void split (midipulse tick)`
Splits the first trigger that brackets the splittick parameter.
- `void split (trigger &trig, midipulse splittick)`
Splits the trigger given by the parameter into two triggers.
- `void grow (midipulse tickfrom, midipulse tickto, midipulse length)`
Grows a trigger.
- `void remove (midipulse tick)`
Deletes the first trigger that brackets the given tick from the trigger-list.
- `bool get_state (midipulse tick)`
Checks the list of triggers against the given tick.
- `bool select (midipulse tick)`
Checks the list of triggers against the given tick.
- `bool unselect ()`
Unselects all triggers.
- `bool intersect (midipulse position, midipulse &start, midipulse &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` (midipulse tick, bool adjustoffset, int which=2)
Moves selected triggers as per the given parameters.
- midipulse `get_selected_start` ()
Gets the selected trigger's start tick.
- midipulse `get_selected_end` ()
Gets the selected trigger's end tick.
- midipulse `get_maximum` ()
Get the ending value of the last trigger in the trigger-list.
- void `move` (midipulse starttick, midipulse distance, bool direction)
Moves triggers in the trigger-list.
- void `copy` (midipulse starttick, midipulse distance)
Not sure what these diagrams are for yet.
- void `clear` ()
Clears the whole list of triggers.
- bool `next` (midipulse *tick_on, midipulse *tick_off, bool *selected, midipulse *tick_offset)
Get the next trigger in the trigger list, and set the parameters based on that trigger.
- trigger `next_trigger` ()
Get the next trigger in the trigger list.
- void `reset_draw_trigger_marker` ()
Sets the draw-trigger iterator to the beginning of the trigger list.

Private Member Functions

- midipulse `adjust_offset` (midipulse 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.
- 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.

12.63.1 Constructor & Destructor Documentation

12.63.1.1 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).
---------------	---

12.63.2 Member Function Documentation

12.63.2.1 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.

FIXED, BEWARE: 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.

Parameters

<i>rhs</i>	Provides the "right-hand side" of the assignment operation.
------------	---

Returns

Returns a reference to self, for use in concatenated assignment operations.

12.63.2.2 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.

12.63.2.3 void seq64::triggers::print (*const std::string* & *seqname*) *const*

Parameters

<i>seqname</i>	A tag name to accompany the print-out, for the human to read.
----------------	---

12.63.2.4 bool seq64::triggers::play (*midipulse* & *start_tick*, *midipulse* & *end_tick*)

This is a new function for [sequence::play\(\)](#) to call.

The for-loop goes through all the triggers, determining if there is are trigger start/end values before the *end_tick*. If so, then the trigger state is set to true (start only within the tick range) or false (end is within the tick range), and the trigger tick is set to start or end. The first start or end trigger that is past the end tick cause the search to end.

If the trigger state has changed, then the start/end ticks are passed back to the sequence, and the trigger offset is adjusted.

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 (trigger turning off), and the caller should stop the playback.

12.63.2.5 `void seq64::triggers::add (midipulse tick, midipulse len, midipulse 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 adjust_offset() first. We think that basically makes sure it is positive.

12.63.2.6 `void seq64::triggers::adjust_offsets_to_length (midipulse newlength)`

Parameters

<i>newlength</i>	Provides the length to which to adjust the offsets.
------------------	---

COMMON CODE?

COMMON CODE?

12.63.2.7 `void seq64::triggers::split (midipulse 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.
------------------	--

12.63.2.8 void seq64::triggers::split (trigger & *trig*, midipulse *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, as a side-effect.
<i>splittick</i>	The position just after where the original trigger will be truncated, and the new trigger begins.

12.63.2.9 void seq64::triggers::grow (midipulse *tickfrom*, midipulse *tickto*, midipulse *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.

12.63.2.10 void seq64::triggers::remove (midipulse *tick*)

Parameters

<i>tick</i>	Provides the tick to be examined.
-------------	-----------------------------------

12.63.2.11 bool seq64::triggers::get_state (midipulse *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.

12.63.2.12 bool seq64::triggers::select (midipulse *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.

12.63.2.13 bool seq64::triggers::unselect ()**Returns**

Always returns false.

12.63.2.14 bool seq64::triggers::intersect (midipulse *position*, midipulse & *start*, midipulse & *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.

12.63.2.15 void seq64::triggers::paste ()

It pastes at the copy end.

12.63.2.16 `bool seq64::triggers::move_selected (midipulse tick, bool fixoffset, int which = 2)`

```

    mintick][0          1][maxtick
                2

```

The `\a` `which` parameter has three possible values:

```

-# 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.

```

Parameters

<i>tick</i>	The tick at which the trigger starts.
<i>fixoffset</i>	Set to true if the offset is to be adjusted.
<i>which</i>	Selects which movement will be done, as discussed above.

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 perfrill window, and keep it from continually incrementing 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.

12.63.2.17 `midipulse 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 `midipulse(-1)` is returned.

12.63.2.18 `midipulse seq64::triggers::get_selected_end ()`

Returns

Returns the `tick_end()` value of the last-selected trigger. If no triggers are selected, then `midipulse(-1)` is returned.

12.63.2.19 `midipulse seq64::triggers::get_maximum ()`

Returns

Returns the tick-end for the last trigger, if available. Otherwise, 0 is returned.

12.63.2.20 `void seq64::triggers::move (midipulse starttick, midipulse distance, bool direction)`

There's no way to optimize this by saving tick values, as they are potentially modified at each step.

Parameters

<i>starttick</i>	The current location of the triggers.
<i>distance</i>	The distance away from the current location to which to move the triggers.
<i>direction</i>	If true, the triggers are moved forward. If false, the triggers are moved backward.

12.63.2.21 void seq64::triggers::copy (midipulse *starttick*, midipulse *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 a point distant from a given tick.

Parameters

<i>starttick</i>	The current location of the triggers.
<i>distance</i>	The distance away from the current location to which to copy the triggers.

12.63.2.22 bool seq64::triggers::next (midipulse * *tick_on*, midipulse * *tick_off*, bool * *selected*, midipulse * *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.

12.63.2.23 `trigger seq64::triggers::next_trigger ()`**Returns**

Returns the next trigger. If there is none, a default trigger object is returned.

12.63.2.24 `midipulse seq64::triggers::adjust_offset (midipulse offset)` `[private]`**Parameters**

<code>offset</code>	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.

12.63.3 Field Documentation**12.63.3.1** `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.

12.63.3.2 `int seq64::triggers::m_length` `[private]`

This might change, we're not yet sure.

12.64 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.

12.64.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

12.64.2 Member Function Documentation

12.64.2.1 void seq64::user_instrument::set_defaults ()

Also invalidates the object.

12.64.2.2 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).

12.64.2.3 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.

12.64.2.4 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.

12.64.2.5 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.

12.64.2.6 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?

12.64.2.7 `void seq64::user_instrument::copy_definitions (const user_instrument & rhs) [private]`

Does not include the validity flag.

12.64.3 Field Documentation

12.64.3.1 `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.

12.64.3.2 `int seq64::user_instrument::m_controller_count [private]`

Often, the "user" configuration file has only a few out of the 128 assigned explicitly.

12.65 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.

12.65.1 Field Documentation

12.65.1.1 `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.

12.65.1.2 `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.

12.65.1.3 `bool seq64::user_instrument_t::controllers_active[SEQ64_MIDI_CONTROLLER_MAX]`

If false, it might be an unsupported controller or a General MIDI device.

12.66 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.

12.66.1 Detailed Description

Will later make the size adjustable, if it makes sense to do so.

12.66.2 Member Function Documentation

12.66.2.1 void seq64::user_midi_bus::set_defaults ()

Also invalidates the object. All 16 of the channels are set to SEQ64_GM_INSTRUMENT_FLAG (-1).

12.66.2.2 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.

12.66.2.3 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).

12.66.2.4 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.

12.66.2.5 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.

12.66.2.6 void seq64::user_midi_bus::copy_definitions (const user_midi_bus & *rhs*) [private]

Does not include the validity flag.

12.66.3 Field Documentation

12.66.3.1 `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.

12.66.3.2 `int seq64::user_midi_bus::m_channel_count` `[private]`

Often, the "user" configuration file has only a few out of the 16 assigned explicitly.

12.67 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.

12.67.1 Field Documentation

12.67.1.1 `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.

12.67.1.2 `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.

12.68 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](#) ()
Default constructor.
- [user_settings](#) (const [user_settings](#) &rhs)
Copy constructor.
- [user_settings](#) & [operator=](#) (const [user_settings](#) &rhs)
Principal assignment operator.
- void [set_defaults](#) ()
Sets the default values.
- void [normalize](#) ()
Calculate the derived values from the already-set values.
- void [set_globals](#) () const
Copies the current values of the member variables into their corresponding global variables.
- void [get_globals](#) ()
Copies the current values of the global variables into their corresponding member variables.
- bool [add_bus](#) (const std::string &alias)
Adds a user bus to the container, but only does so if the name parameter is not empty.
- bool [add_instrument](#) (const std::string &instname)
Adds a user instrument to the container, but only does so if the name parameter is not empty.
- const [user_midi_bus](#) & [bus](#) (int index)
'Getter' function for member Unlike the non-const version this function is public.
- const [user_instrument](#) & [instrument](#) (int index)
'Getter' function for member Unlike the non-const version this function is public.
- int [bus_count](#) () const
'Getter' function for member m_midi_buses.size()
- void [set_bus_instrument](#) (int index, int channel, int instrum)
'Getter' function for member m_midi_buses[index].instrument[channel] Currently this function is used, in the [userfile::parse\(\)](#) function.
- int [bus_instrument](#) (int buss, int channel)
'Getter' function for member m_midi_buses[buss].instrument[channel]
- const std::string & [bus_name](#) (int buss)
'Getter' function for member m_midi_buses[buss].name
- int [instrument_count](#) () const
'Getter' function for member m_instruments.size()
- void [set_instrument_controllers](#) (int index, int cc, const std::string &ccname, bool isactive)
'Setter' function for member m_midi_instrument_defs[index].controllers, controllers_active
- const std::string & [instrument_name](#) (int instrum)
'Getter' function for member m_instruments[instrument].instrument (name of instrument)
- 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*
- bool [allow_two_perfedits](#) () const
 - 'Getter' function for member m_allow_two_perfedits*
- int [perf_h_page_increment](#) () const
 - 'Getter' function for member m_h_perf_page_increment*
- int [perf_v_page_increment](#) () const
 - 'Getter' function for member m_v_perf_page_increment*
- bool [progress_bar_colored](#) () const
 - 'Getter' function for member m_progress_bar_colored*
- bool [progress_bar_thick](#) () const
 - 'Getter' function for member m_progress_bar_thick*
- int [window_redraw_rate](#) () const
 - 'Getter' function for member m_window_redraw_rate_ms*
- 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 [use_new_font](#) (bool flag)
'Setter' function for member m_use_new_font
- void [allow_two_perfedits](#) (bool flag)
Sets the value of allowing two perfedits to be created and shown to the user.
- void [perf_h_page_increment](#) (int inc)
Sets the horizontal page increment size for the horizontal scrollbar of a perfedit window.
- void [perf_v_page_increment](#) (int inc)
Sets the vertical page increment size for the vertical scrollbar of a perfedit window.
- void [progress_bar_colored](#) (bool flag)
'Setter' function for member m_progress_bar_colored
- void [progress_bar_thick](#) (bool flag)
'Setter' function for member m_progress_bar_thick
- void [window_redraw_rate](#) (int ms)
'Setter' function for member m_window_redraw_rate_ms
- 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 crampedness 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 crampedness 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

- 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.*

 - bool [m_allow_two_perfedits](#)
- Enables the usage of two perfedit windows, for added convenience in editing multi-set songs.*

 - int [m_h_perf_page_increment](#)
- Allows a changed to the page size for the horizontal scroll bar.*

 - int [m_v_perf_page_increment](#)
- Allows a changed to the page size for the vertical scroll bar.*

 - bool [m_progress_bar_colored](#)
- If set, makes progress bars have the "progress_color()", instead of black.*

 - bool [m_progress_bar_thick](#)
- If set, makes progress bars thicker than 1 pixel...*

 - int [m_window_redraw_rate_ms](#)
- Provides the global setting for redraw rate of windows.*

 - 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.

12.68.1 Detailed Description

These settings will eventually be made part of the "user" settings file.

12.68.2 Member Typedef Documentation

12.68.2.1 `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.

12.68.2.2 `typedef std::vector<user_instrument> seq64::user_settings::Instruments` [private]

Internal type for the container of [user_instrument](#) objects.

12.68.3 Member Enumeration Documentation

12.68.3.1 `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.

12.68.4 Member Function Documentation

12.68.4.1 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!

12.68.4.2 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!!!!

12.68.4.3 void seq64::user_settings::get_globals ()

Should be called before settings are written to the "user" configuration file.

12.68.4.4 const user_midi_bus& seq64::user_settings::bus (int *index*) [inline]

Cannot append the const specifier.

12.68.4.5 const user_instrument& seq64::user_settings::instrument (int *index*) [inline]

Cannot append the const specifier.

12.68.4.6 bool seq64::user_settings::controller_active (int *buss*, int *channel*, int *cc*) [inline]

It also has a shorter name.

12.68.4.7 const std::string& seq64::user_settings::controller_name (int *buss*, int *channel*, int *cc*) [inline]

It also has a shorter name.

12.68.4.8 void seq64::user_settings::zoom (int *value*)

The default value is 2.

12.68.4.9 void seq64::user_settings::mainwnd_rows (int *value*) [protected]

The default value is 4. Dependent values are recalculated after the assignment.

12.68.4.10 void seq64::user_settings::mainwnd_cols (int *value*) [protected]

The default value is 8. Dependent values are recalculated after the assignment.

12.68.4.11 void seq64::user_settings::max_sets (int *value*) [protected]

The default value is 32. Dependent values are recalculated after the assignment.

12.68.4.12 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.

12.68.4.13 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.

12.68.4.14 void seq64::user_settings::mainwid_border (int *value*) [protected]

The default value is 0. Dependent values are recalculated after the assignment.

12.68.4.15 void seq64::user_settings::mainwid_spacing (int *value*) [protected]

The default value is 2. Dependent values are recalculated after the assignment.

12.68.4.16 void seq64::user_settings::control_height (int *value*) [protected]

The default value is 0. Dependent values are recalculated after the assignment.

12.68.4.17 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.

12.68.4.18 void seq64::user_settings::perf_h_page_increment (int *inc*)

This value ranges from 1 (the original value, really too small for a "page" operation) to 6 (which is 24 measures, the same as the typical width of the perfolli)

12.68.4.19 `void seq64::user_settings::perf_v_page_increment (int inc)`

This value ranges from 1 (the original value, really too small for a "page" operation) to 18 (which is 18 tracks, slightly more than the typical height of the perfroll)

12.68.4.20 `void seq64::user_settings::midi_ppqn (int value)`

The default value is 192. Dependent values may be recalculated after the assignment.

12.68.4.21 `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".)

12.68.4.22 `void seq64::user_settings::midi_beats_per_bar (int value)` `[protected]`

The default value is 4.

12.68.4.23 `void seq64::user_settings::midi_beats_per_minute (int value)` `[protected]`

The default value is 120.

12.68.4.24 `void seq64::user_settings::midi_beat_width (int bw)` `[protected]`

The default value is 4.

12.68.4.25 `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.

12.68.4.26 `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.

12.68.5 Field Documentation

12.68.5.1 `Busses seq64::user_settings::m_midi_buses` `[private]`

Since this object is a vector, its size is adjustable.

12.68.5.2 Instruments seq64::user_settings::m_instruments [private]

The size is adjustable, and grows as objects are added.

12.68.5.3 mainwid_grid_style_t seq64::user_settings::m_grid_style [private]

These are not labelled, but are present in the "user" configuration file in the following order:

```
-# grid-style
-# grid-brackets
-# mainwnd-rows
-# mainwnd-cols
-# max-set
-# mainwid-border
-# control-height
-# zoom
-# global-seq-feature
-# use-new-font
-# allow-two-perfedit
-# perf-h-page-increment
-# perf-v-page-increment
-# progress-bar-colored (new)
-# progress-bar-thick (new)
-# window-redraw-rate-ms (new)
```

Specifies the current grid style.

12.68.5.4 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.

12.68.5.5 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 Sequencer64 deserve the 64 in its name.

12.68.5.6 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".

12.68.5.7 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.

12.68.5.8 `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.

12.68.5.9 `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.

12.68.5.10 `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.

12.68.5.11 `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`.

12.68.5.12 `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`.

12.68.5.13 `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`.

12.68.5.14 `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.

12.68.5.15 `bool seq64::user_settings::m_allow_two_perfedits` `[private]`

Defaults to true.

12.68.5.16 `int seq64::user_settings::m_h_perf_page_increment` `[private]`

The value used to be hardwired to 1 (in four-measure units), now it defaults to 4 (16 measures at a time). The value of 1 is already covered by the scrollbar arrows.

12.68.5.17 `int seq64::user_settings::m_v_perf_page_increment` `[private]`

The value used to be hardwired to 1 (in single-track units), now it defaults to 8. The value of 1 is already covered by the scrollbar arrows.

12.68.5.18 `bool seq64::user_settings::m_progress_bar_colored` `[private]`

This value is hardwired in the `gui_palette_gtk2` module, to red. Really, that is the only color that stands out as well as black.

12.68.5.19 `bool seq64::user_settings::m_progress_bar_thick` `[private]`

2 pixels. It isn't useful to support anything thicker.

12.68.5.20 `int seq64::user_settings::m_window_redraw_rate_ms` `[private]`

Not all windows use this yet. The default is 40 ms (`c_redraw_ms`, which is 20 ms in Windows builds)), but some windows originally used 25 ms, so beware of side-effects.

12.68.5.21 `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.

12.68.5.22 `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.

12.68.5.23 `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`.

12.68.5.24 `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 `SEQ64_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.

12.68.5.25 `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.

12.68.5.26 `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.

12.68.5.27 `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`.

12.68.5.28 `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.

12.68.5.29 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.

12.68.5.30 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;
```

12.68.5.31 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.

12.68.5.32 int seq64::user_settings::m_seqarea_seq_x [private]

These are used only in the mainwid module.

12.68.5.33 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  
);
```

12.68.5.34 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.

12.68.5.35 const int seq64::user_settings::mc_min_zoom [private]

It's value is 1.

12.68.5.36 `const int seq64::user_settings::mc_max_zoom [private]`

It's value was 32, but is now 128, to allow for better presentation of high PPQN valued sequences.

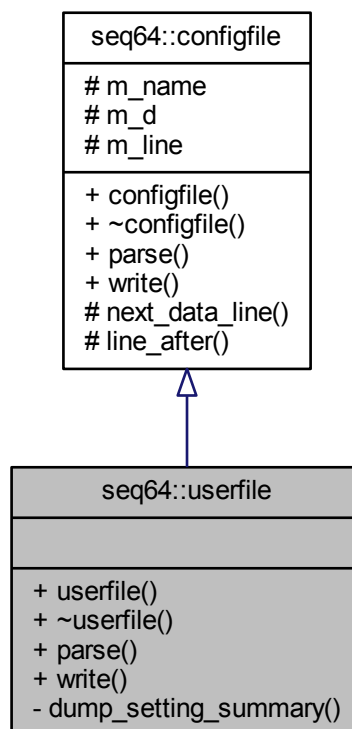
12.68.5.37 `const int seq64::user_settings::mc_baseline_ppqn [private]`

This value is necessary in order to keep user-interface elements stable when different PPQNs are used. It is set to DEFAULT_PPQN.

12.69 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

12.69.1 Member Function Documentation

12.69.1.1 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](#).

12.69.1.2 bool seq64::userfile::write (const [perform](#) & *a_perf*) [virtual]

Parameters

<i>a_perf</i>	The performance object, currently unused.
---------------	---

Implements [seq64::configfile](#).

12.69.1.3 void seq64::userfile::dump_setting_summary () [private]

Does work only if PLATFORM_DEBUG is defined; see the [user_settings](#) class.

Index

- ~event
 - seq64::event, [96](#)
- ~eventedit
 - seq64::eventedit, [112](#)
- ~jack_assistant
 - seq64::jack_assistant, [150](#)
- ~keys_perform
 - seq64::keys_perform, [161](#)
- ~keys_perform_gtk2
 - seq64::keys_perform_gtk2, [165](#)
- ~mastermidibus
 - seq64::mastermidibus, [194](#)
- ~perfedit
 - seq64::perfedit, [237](#)
- ~perform
 - seq64::perform, [251](#)
- ~perfroll
 - seq64::perfroll, [275](#)
- ~seqmenu
 - seq64::seqmenu, [319](#)
- ~seqroll
 - seq64::seqroll, [326](#)
- about_dialog
 - seq64::mainwnd, [187](#)
- add
 - seq64::editable_events, [90](#)
 - seq64::event_list, [105](#)
 - seq64::triggers, [365](#)
- add_event
 - seq64::sequence, [345](#), [351](#)
- add_long
 - seq64::midi_container, [201](#)
- add_note
 - seq64::sequence, [351](#)
- add_sequence
 - seq64::perform, [253](#)
- add_trigger
 - seq64::midifile, [222](#)
 - seq64::sequence, [345](#)
- add_variable
 - seq64::midi_container, [201](#)
- adj_callback_bpm
 - seq64::mainwnd, [188](#)
- adj_callback_ss
 - seq64::mainwnd, [188](#)
- adjust_offset
 - seq64::triggers, [370](#)
- adjust_offsets_to_length
 - seq64::triggers, [365](#)
- adjust_trigger_offsets_to_length
 - seq64::sequence, [358](#)
- adjustment_dummy
 - seq64, [65](#)
- alias
 - seq64::user_midi_bus_t, [376](#)
- all_notes_off
 - seq64::perform, [257](#)
- analyze
 - seq64::editable_event, [85](#)
- any_selected_notes
 - seq64::event_list, [108](#)
- append_sysex
 - seq64::event, [100](#)
- apply_length
 - seq64::seqedit, [302](#)
- BLACK_ON_CYAN
 - seq64::font, [129](#)
- BLACK_ON_YELLOW
 - seq64::font, [129](#)
- BLACK
 - seq64::font, [129](#)
- background_sequence
 - seq64::sequence, [356](#)
- beats_per_minute_from_tempo
 - seq64, [55](#)
- bus
 - seq64::user_settings, [384](#)
- Busses
 - seq64::user_settings, [383](#)
- button
 - seq64::options, [230](#)
- c_chord_text
 - seq64, [68](#)
- c_controller_names
 - seq64, [65](#)
- c_mainwid_x
 - seq64, [69](#)
- c_max_instruments
 - seq64, [68](#)
- c_midi_track_ctrl
 - seq64, [67](#)
- c_midibus
 - seq64, [66](#)
- c_midibus_output_size
 - seq64, [66](#)
- c_music_scales
 - seq64, [49](#)

- c_scales_policy
 - seq64, [67](#)
- c_scales_transpose_dn_neg
 - seq64, [68](#)
- c_scales_transpose_up
 - seq64, [67](#)
- c_select_all_notes
 - seq64, [69](#)
- CYAN_ON_BLACK
 - seq64::font, [129](#)
- calculate_base_sizes
 - seq64::mainwid, [180](#)
- category
 - seq64::editable_event, [84](#)
- category_channel_message
 - seq64::editable_event, [82](#)
- category_meta_event
 - seq64::editable_event, [82](#)
- category_name
 - seq64::editable_event, [82](#)
- category_prop_event
 - seq64::editable_event, [82](#)
- category_system_message
 - seq64::editable_event, [82](#)
- category_t
 - seq64::editable_event, [82](#)
- change_event_data_range
 - seq64::sequence, [352](#)
- change_horz
 - seq64::seqevent, [310](#)
 - seq64::seqroll, [328](#)
- change_vert
 - seq64::eventslots, [124](#)
- channel_count
 - seq64::user_midi_bus, [375](#)
- channel_max
 - seq64::user_midi_bus, [375](#)
- CharList
 - seq64::midi_list, [203](#)
- check_channel
 - seq64::event, [97](#)
- checklen
 - seq64::midifile, [222](#)
- choose_ppqn
 - seq64, [62](#)
- clamp_track
 - seq64::perform, [268](#)
- clear
 - seq64::event_list, [106](#)
- clear_all
 - seq64::perform, [252](#)
- clear_sequence_triggers
 - seq64::perform, [253](#)
- clear_triggers
 - seq64::sequence, [349](#)
- clear_window
 - seq64::gui_drawingarea_gtk2, [137](#)
- click
 - seq64::click, [73](#), [74](#)
- client_open
 - seq64::jack_assistant, [153](#)
- clock
 - seq64::mastermidibus, [195](#)
 - seq64::midibus, [214](#)
- clock_e
 - seq64, [49](#)
- clock_tick_duration_bogus
 - seq64, [57](#)
- clock_ticks_from_ppqn
 - seq64, [57](#)
- collapse
 - seq64::perfedit, [238](#)
- Color
 - seq64::font, [129](#)
- cond
 - seq64::condition_var, [76](#)
- configfile
 - seq64::configfile, [77](#)
- continue_from
 - seq64::mastermidibus, [195](#)
 - seq64::midibus, [214](#)
- control_height
 - seq64::user_settings, [385](#)
- controller_active
 - seq64::user_instrument, [372](#)
 - seq64::user_settings, [384](#)
- controller_max
 - seq64::user_instrument, [372](#)
- controller_name
 - seq64::user_instrument, [372](#)
 - seq64::user_settings, [384](#)
- controllers
 - seq64::user_instrument_t, [373](#)
- controllers_active
 - seq64::user_instrument_t, [373](#)
- convert_t
 - seq64::seqevent, [310](#)
- convert_tn
 - seq64::seqroll, [328](#)
- convert_x
 - seq64::perfroll, [277](#)
 - seq64::seqevent, [310](#)
- convert_xy
 - seq64::perfroll, [277](#)
- convert_y
 - seq64::eventslots, [123](#)
- copy
 - seq64::perfedit, [238](#)
 - seq64::triggers, [369](#)
- copy_definitions
 - seq64::user_instrument, [372](#)
 - seq64::user_midi_bus, [375](#)
- copy_events
 - seq64::sequence, [357](#)
- copy_selected
 - seq64::sequence, [350](#)

- copy_triggers
 - seq64::perform, [254](#)
 - seq64::sequence, [349](#)
- count
 - seq64::editable_events, [90](#)
 - seq64::event_list, [105](#)
- count_selected_events
 - seq64::event_list, [108](#)
- create_lash_driver
 - seq64, [64](#)
- create_menus
 - seq64::seqedit, [303](#)
- cut_selected
 - seq64::sequence, [351](#)
- DRAW_FIN
 - seq64, [49](#)
- decrement_beats_per_minute
 - seq64::perform, [263](#)
- decrement_bottom
 - seq64::eventslots, [125](#)
- decrement_current
 - seq64::eventslots, [125](#)
- decrement_selected
 - seq64::sequence, [353](#)
- decrement_top
 - seq64::eventslots, [125](#)
- deinit
 - seq64::jack_assistant, [150](#)
- deinit_in
 - seq64::midibus, [213](#)
- deinit_jack
 - seq64::perform, [266](#)
- del_trigger
 - seq64::sequence, [346](#)
- delete_current_event
 - seq64::eventslots, [121](#)
- delete_lash_driver
 - seq64, [64](#)
- delete_sequence
 - seq64::perform, [253](#)
- delta_time_us_to_ticks
 - seq64, [56](#)
- do_action
 - seq64::seqedit, [304](#)
- double_ticks_from_ppqn
 - seq64, [57](#)
- draw_background
 - seq64::seqevent, [309](#)
- draw_drawable_row
 - seq64::perfroll, [278](#)
- draw_event
 - seq64::eventslots, [123](#)
- draw_events
 - seq64::eventslots, [124](#)
- draw_events_on
 - seq64::seqevent, [310](#)
 - seq64::seqroll, [328](#)
- draw_events_on_pixmap
 - seq64::seqroll, [327](#)
- draw_key
 - seq64::seqkeys, [315](#)
- draw_line
 - seq64::gui_drawingarea_gtk2, [137–139](#)
- draw_line_on_pixmap
 - seq64::gui_drawingarea_gtk2, [138](#)
- draw_marker_on_sequence
 - seq64::mainwid, [178](#)
- draw_normal_rectangle_on_pixmap
 - seq64::gui_drawingarea_gtk2, [142](#)
- draw_pixmap_on_window
 - seq64::seqevent, [310](#)
- draw_progress
 - seq64::perfroll, [276](#)
- draw_progress_on_window
 - seq64::seqroll, [327](#)
- draw_rectangle
 - seq64::gui_drawingarea_gtk2, [140, 141](#)
- draw_rectangle_on_pixmap
 - seq64::gui_drawingarea_gtk2, [142](#)
- draw_sequence
 - seq64::perfnames, [242](#)
- draw_sequence_on
 - seq64::perfroll, [277](#)
- draw_sequence_on_pixmap
 - seq64::mainwid, [178](#)
- draw_sequence_pixmap_on_window
 - seq64::mainwid, [179](#)
- draw_sequences_on_pixmap
 - seq64::mainwid, [179](#)
- draw_type
 - seq64, [49](#)
- drop_event
 - seq64::seqevent, [310](#)
- dump_setting_summary
 - seq64::userfile, [393](#)
- dump_summary
 - seq64::user_settings, [385](#)
- e_clock_mod
 - seq64, [49](#)
- e_clock_off
 - seq64, [49](#)
- e_clock_pos
 - seq64, [49](#)
- e_deselect
 - seq64::sequence, [341](#)
- e_is_selected
 - seq64::sequence, [341](#)
- e_jack_connect
 - seq64::options, [230](#)
- e_jack_disconnect
 - seq64::options, [230](#)
- e_jack_master
 - seq64::options, [230](#)
- e_jack_master_cond
 - seq64::options, [230](#)
- e_jack_start_mode_live

- seq64::options, 230
- e_jack_start_mode_song
 - seq64::options, 230
- e_jack_transport
 - seq64::options, 230
- e_remove_one
 - seq64::sequence, 341
- e_select
 - seq64::sequence, 341
- e_select_one
 - seq64::sequence, 341
- e_toggle_selection
 - seq64::sequence, 341
- e_would_select
 - seq64::sequence, 341
- EVENT_MIDI_SYSEX
 - seq64, 65
- EVENT_NOTE_OFF
 - seq64, 65
- EVENT_NULL_CHANNEL
 - seq64, 66
- edit_callback_notepad
 - seq64::mainwnd, 188
- editable_event
 - seq64::editable_event, 83
- editable_events
 - seq64::editable_events, 89
- enqueue_draw
 - seq64::perfedit, 237
 - seq64::perfnames, 242
 - seq64::perfroll, 278
 - seq64::perftime, 283
- enregister_peer
 - seq64::perfedit, 237
- errdump
 - seq64::midifile, 226
- error_message
 - seq64::jack_assistant, 153
- event
 - seq64::event, 96
- event_count
 - seq64::sequence, 341
- event_key
 - seq64::event_list::event_key, 102
- event_list
 - seq64::event_list, 105
- eventedit
 - seq64::eventedit, 111
- events
 - seq64::keybindentry, 158
- expand
 - seq64::perfedit, 238
- extract_timing_numbers
 - seq64, 50
- file_accessible
 - seq64, 61
- file_executable
 - seq64, 61
- file_exists
 - seq64, 60
- file_import_dialog
 - seq64::mainwnd, 187
- file_is_directory
 - seq64, 61
- file_readable
 - seq64, 60
- file_writable
 - seq64, 60
- fill
 - seq64::midi_container, 200
- fill_background_pixmap
 - seq64::perfroll, 276
- fill_container
 - seq64::sequence, 356
- fill_top_bar
 - seq64::seqedit, 303
- finish
 - seq64::perform, 253
- flush
 - seq64::mastermidibus, 195
- follow_progress
 - seq64::seqroll, 327
- font_render
 - seq64, 65
- format_timestamp
 - seq64::editable_event, 85
- FruityPerfInput
 - seq64::perfroll, 279
- get
 - seq64::midi_container, 201
 - seq64::midi_list, 203
 - seq64::midi_vector, 210
- get_beat_width
 - seq64::sequence, 342
- get_clipboard_box
 - seq64::sequence, 351
- get_clock
 - seq64::mastermidibus, 198
- get_data
 - seq64::event, 100
- get_globals
 - seq64::user_settings, 384
- get_group_mute_state
 - seq64::perform, 261
- get_input
 - seq64::mastermidibus, 198
- get_keys
 - seq64::keys_perform, 162
- get_last_tick
 - seq64::sequence, 342
- get_max_tick
 - seq64::perform, 254
- get_max_trigger
 - seq64::perform, 255
 - seq64::sequence, 349
- get_maximum

- seq64::triggers, 368
- get_measures
 - seq64::seqedit, 302
- get_midi_event
 - seq64::mastermidibus, 196
- get_midi_in_bus_name
 - seq64::mastermidibus, 195
- get_midi_out_bus_name
 - seq64::mastermidibus, 194
- get_minmax_note_events
 - seq64::sequence, 355
- get_name
 - seq64::sequence, 342
- get_next_event
 - seq64::sequence, 356
- get_next_note_event
 - seq64::sequence, 355
- get_num_selected_events
 - seq64::sequence, 350
- get_num_selected_notes
 - seq64::sequence, 350
- get_rank
 - seq64::event, 101
- get_screen_set_notepad
 - seq64::perform, 255
- get_selected_box
 - seq64::sequence, 351
- get_selected_end
 - seq64::triggers, 368
- get_selected_start
 - seq64::triggers, 368
- get_sequence
 - seq64::perform, 259
- get_state
 - seq64::triggers, 366
- get_trigger_state
 - seq64::sequence, 346
- grid_style_black
 - seq64::user_settings, 383
- grid_style_max
 - seq64::user_settings, 383
- grid_style_normal
 - seq64::user_settings, 383
- grid_style_white
 - seq64::user_settings, 383
- groups
 - seq64::keybindentry, 158
- grow
 - seq64::perfedit, 238
 - seq64::triggers, 366
- grow_selected
 - seq64::sequence, 353
- grow_trigger
 - seq64::sequence, 346
- gui_assistant
 - seq64::gui_assistant, 132
- gui_palette_gtk2
 - seq64::gui_palette_gtk2, 145
- gui_window_gtk2
 - seq64::gui_window_gtk2, 147
- handle_config
 - seq64::lash, 169
- handle_event
 - seq64::lash, 169
- handle_insert
 - seq64::eventedit, 114
- handle_midi_control
 - seq64::perform, 255
- handle_modify
 - seq64::eventedit, 114
- handle_motion_key
 - seq64::Seq24PerfInput, 288
- handle_save
 - seq64::eventedit, 114
- help_check
 - seq64, 58
- highlight
 - seq64::perform, 264
- home_config_directory
 - seq64::rc_settings, 286
- horizontal_adjust
 - seq64::seqedit, 301
 - seq64::seqroll, 327
- idle_progress
 - seq64::maintime, 172
- idle_redraw
 - seq64::seqdata, 294
 - seq64::sequevent, 310
- increment
 - seq64::midi_splitter, 206
- increment_beats_per_minute
 - seq64::perform, 264
- increment_bottom
 - seq64::eventslots, 126
- increment_current
 - seq64::eventslots, 125
- increment_selected
 - seq64::sequence, 353
- increment_top
 - seq64::eventslots, 125
- info_message
 - seq64::jack_assistant, 152
- init
 - seq64::font, 129
 - seq64::jack_assistant, 150
 - seq64::lash, 169
 - seq64::mastermidibus, 194
- init_before_show
 - seq64::perfedit, 237
 - seq64::perfroll, 276
- init_clock
 - seq64::mastermidibus, 196
 - seq64::midibus, 214
- init_in
 - seq64::midibus, 213

- init_in_sub
 - seq64::midibus, 214
- init_jack
 - seq64::perform, 266
- init_out
 - seq64::midibus, 213
- init_out_sub
 - seq64::midibus, 213
- inner_start
 - seq64::perform, 268
- inner_stop
 - seq64::perform, 268
- insert_event
 - seq64::eventslots, 120, 121
- install_sequence
 - seq64::perform, 267
- instrument
 - seq64::user_instrument_t, 373
 - seq64::user_midi_bus, 375
 - seq64::user_midi_bus_t, 376
 - seq64::user_settings, 384
- Instruments
 - seq64::user_settings, 383
- interaction_method_t
 - seq64, 49
- intersect
 - seq64::triggers, 367
- intersect_events
 - seq64::sequence, 347
- intersect_notes
 - seq64::sequence, 347
- intersect_triggers
 - seq64::sequence, 347
- is_active
 - seq64::perform, 259
- is_channel_msg
 - seq64::event, 97
- is_desired_cc_or_not_cc
 - seq64::event, 98
- is_dirty_edit
 - seq64::perform, 258
 - seq64::sequence, 344
- is_dirty_main
 - seq64::perform, 258
 - seq64::sequence, 344
- is_dirty_names
 - seq64::perform, 258
 - seq64::sequence, 344
- is_dirty_perf
 - seq64::perform, 258
 - seq64::sequence, 344
- is_edit_sequence
 - seq64::seqmenu, 320
- is_letter
 - seq64::keystroke, 167
- is_midi_control_valid
 - seq64::perform, 266
- is_more_input
 - seq64::mastermidibus, 196
- is_mseq_valid
 - seq64::perform, 267
- is_note_msg
 - seq64::event, 98
- is_one_byte_msg
 - seq64::event, 98
- is_screenset_valid
 - seq64::perform, 266
- is_seq_valid
 - seq64::perform, 267
- is_sysex_special_id
 - seq64::midifile, 226
- is_two_byte_msg
 - seq64::event, 98
- jack_assistant
 - seq64::jack_assistant, 150
- jack_idle_connect
 - seq64::gui_assistant_gtk2, 133
- jack_session_callback
 - seq64, 63
 - seq64::jack_assistant, 157
- jack_shutdown_callback
 - seq64, 63
 - seq64::jack_assistant, 155
- jack_sync_callback
 - seq64, 62
 - seq64::jack_assistant, 156
 - seq64::perform, 269
- jack_timebase_callback
 - seq64, 63
 - seq64::jack_assistant, 156
- Key
 - seq64::editable_events, 89
- key_name
 - seq64::keys_perform, 162
 - seq64::keys_perform_gtk2, 165
- key_press_event
 - seq64::perftime, 283
- keybindentry
 - seq64::keybindentry, 158
- keystroke
 - seq64::keystroke, 166, 167
- lash
 - seq64::lash, 168
- lash_driver
 - seq64, 64
- lash_timeout_connect
 - seq64::gui_assistant_gtk2, 133
- launch
 - seq64::perform, 252
- launch_input_thread
 - seq64::perform, 265
- launch_output_thread
 - seq64::perform, 265
- line_after

- seq64::configfile, 78
- line_color
 - seq64::gui_palette_gtk2, 145
- link_new
 - seq64::event_list, 107
 - seq64::sequence, 354
- load_events
 - seq64::editable_events, 89
 - seq64::eventslots, 120
- location
 - seq64::keybindentry, 158
- log2_time_sig_value
 - seq64, 54
- long_options
 - seq64, 69
- lookup_keyevent_key
 - seq64::perform, 262
- m_32nds_per_quarter
 - seq64::sequence, 359
- m_adjust_load_offset
 - seq64::mainwnd, 191
- m_allow_two_perfedits
 - seq64::user_settings, 388
- m_b_on_c_pixmap
 - seq64::font, 130
- m_b_on_y_pixmap
 - seq64::font, 130
- m_background
 - seq64::gui_drawingarea_gtk2, 143
- m_background_sequence
 - seq64::sequence, 360
- m_bar_width
 - seq64::maintime, 173
- m_beat_width
 - seq64::maintime, 173
 - seq64::midi_timing, 209
 - seq64::perform, 270
- m_beats_per_bar
 - seq64::perform, 270
- m_beats_per_measure
 - seq64::midi_timing, 208
- m_beats_per_minute
 - seq64::mastermidibus, 198
 - seq64::midi_timing, 208
- m_black_pixmap
 - seq64::font, 130
- m_box_height
 - seq64::maintime, 173
- m_box_width
 - seq64::maintime, 173
- m_bpm
 - seq64::perfedit, 239
- m_button
 - seq64::click, 74
- m_button_down
 - seq64::mainwid, 182
- m_button_play
 - seq64::mainwnd, 191
- m_bw
 - seq64::perfedit, 239
- m_c_on_b_pixmap
 - seq64::font, 130
- m_category
 - seq64::editable_event, 87
- m_channel
 - seq64::event, 101
- m_channel_count
 - seq64::user_midi_bus, 376
- m_char_list
 - seq64::midifile, 227
- m_char_w
 - seq64::eventslots, 127
 - seq64::perfnames, 243
- m_clip_mask
 - seq64::font, 130
- m_clock_mod
 - seq64::midibus, 215
- m_clocks_per_metronome
 - seq64::sequence, 359
- m_control_height
 - seq64::user_settings, 388
- m_controller_count
 - seq64::user_instrument, 373
- m_current_event
 - seq64::editable_events, 91
- m_current_index
 - seq64::eventslots, 127
- m_data
 - seq64::event, 101
 - seq64::midifile, 227
- m_disable_reported
 - seq64::midifile, 227
- m_divisions
 - seq64::midi_measures, 204
- m_drop_x
 - seq64::gui_drawingarea_gtk2, 143
- m_error_is_fatal
 - seq64::midifile, 227
- m_error_message
 - seq64::midifile, 227
- m_file_size
 - seq64::midifile, 227
- m_flash_height
 - seq64::maintime, 173
- m_flash_width
 - seq64::maintime, 173
- m_font_h
 - seq64::font, 130
- m_font_w
 - seq64::font, 129
- m_foreground
 - seq64::gui_drawingarea_gtk2, 143
- m_format_timestamp
 - seq64::editable_event, 87
- m_fruity_interaction
 - seq64::perfroll, 279

- m_global_seq_feature_save
 - seq64::user_settings, 388
- m_gmute_tracks
 - seq64::user_settings, 390
- m_grid_brackets
 - seq64::user_settings, 387
- m_grid_style
 - seq64::user_settings, 387
- m_h_page_increment
 - seq64::perftroll, 279
- m_h_perf_page_increment
 - seq64::user_settings, 389
- m_has_link
 - seq64::event, 101
- m_initial_snap
 - seq64::seqedit, 305
- m_instruments
 - seq64::user_settings, 386
- m_is_modified
 - seq64::event_list, 108
 - seq64::perform, 271
- m_is_press
 - seq64::keystroke, 167
- m_is_valid
 - seq64::user_instrument, 373
 - seq64::user_midi_bus, 376
- m_key
 - seq64::keybindentry, 159
 - seq64::keystroke, 167
- m_key_bpm_up
 - seq64::keys_perform, 163
- m_key_show_ui_sequence_number
 - seq64::keys_perform, 163
- m_left_marker_tick
 - seq64::perftime, 283
- m_left_tick
 - seq64::perform, 270
- m_length
 - seq64::sequence, 359
 - seq64::triggers, 370
- m_line
 - seq64::configfile, 78
- m_main_wid
 - seq64::mainwnd, 191
- m_mainperf
 - seq64::gui_drawingarea_gtk2, 143
- m_mainwid_border
 - seq64::user_settings, 387
- m_mainwid_x
 - seq64::user_settings, 391
- m_mainwnd_cols
 - seq64::user_settings, 387
- m_mainwnd_rows
 - seq64::user_settings, 387
- m_max_sequence
 - seq64::user_settings, 391
- m_max_sets
 - seq64::perform, 270
- seq64::user_settings, 387
- m_measure_length
 - seq64::perftime, 283
- m_midi_beat_width
 - seq64::user_settings, 390
- m_midi_beats_per_measure
 - seq64::user_settings, 390
- m_midi_beats_per_minute
 - seq64::user_settings, 390
- m_midi_buses
 - seq64::user_settings, 386
- m_midi_buss_override
 - seq64::user_settings, 390
- m_midi_channel
 - seq64::sequence, 358
- m_midi_parameters
 - seq64::editable_events, 91
- m_midi_ppqn
 - seq64::user_settings, 390
- m_modified
 - seq64::seqmenu, 321
- m_modifier
 - seq64::click, 74
 - seq64::keystroke, 168
- m_moving_seq
 - seq64::mainwid, 182
- m_musical_key
 - seq64::sequence, 360
- m_musical_scale
 - seq64::sequence, 360
- m_mutex
 - seq64::sequence, 360
- m_name_meta
 - seq64::editable_event, 87
- m_name_status
 - seq64::editable_event, 87
- m_namebox_w
 - seq64::perfnames, 243
- m_names_chars
 - seq64::perfnames, 242
- m_names_y
 - seq64::perfnames, 243
- m_new_format
 - seq64::midifile, 227
- m_notebook
 - seq64::options, 230
- m_number_h
 - seq64::seqdata, 296
- m_number_offset_y
 - seq64::seqdata, 296
- m_number_w
 - seq64::seqdata, 296
- m_one_measure
 - seq64::perform, 270
- m_parent
 - seq64::editable_event, 86
 - seq64::perfnames, 242
 - seq64::perftroll, 279

- seq64::perftime, 283
- seq64::sequence, 358
- m_pixmap
 - seq64::font, 130
 - seq64::gui_drawingarea_gtk2, 143
- m_playback_mode
 - seq64::perform, 269
- m_playing_notes
 - seq64::sequence, 359
- m_playscreen_offset
 - seq64::perform, 269
- m_pos
 - seq64::midifile, 227
 - seq64::seqedit, 306
 - seq64::seqroll, 329
- m_ppqn
 - seq64::maintime, 174
 - seq64::mainwnd, 191
 - seq64::midi_timing, 209
 - seq64::triggers, 370
- m_progress_bar_colored
 - seq64::user_settings, 389
- m_progress_bar_thick
 - seq64::user_settings, 389
- m_raise
 - seq64::sequence, 359
- m_rank
 - seq64::event_list::event_key, 103
- m_redraw_period_ms
 - seq64::gui_window_gtk2, 147
- m_right_marker_tick
 - seq64::perftime, 284
- m_right_tick
 - seq64::perform, 270
- m_save_user_config
 - seq64::user_settings, 391
- m_screenset_offset
 - seq64::mainwid, 182
- m_screenset_slots
 - seq64::mainwid, 182
- m_seq_number
 - seq64::sequence, 359
- m_seqarea_seq_x
 - seq64::user_settings, 391
- m_seqarea_x
 - seq64::user_settings, 391
- m_seqchars_x
 - seq64::user_settings, 389
- m_seqdata_wid
 - seq64::seqedit, 306
- m_seqedit
 - seq64::seqmenu, 321
- m_seqedit_bgsequence
 - seq64::user_settings, 388
- m_seqedit_key
 - seq64::user_settings, 388
- m_seqedit_scale
 - seq64::user_settings, 388
- m_seqkeys_wid
 - seq64::seqedit, 306
- m_seqs
 - seq64::perform, 269
- m_seqs_in_set
 - seq64::perform, 270
 - seq64::user_settings, 390
- m_seqtime_wid
 - seq64::seqedit, 306
- m_sequence
 - seq64::editable_events, 91
- m_sequence_count
 - seq64::perform, 271
- m_sequence_max
 - seq64::perform, 271
- m_setbox_w
 - seq64::eventslots, 127
 - seq64::perfnames, 243
- m_sigpipe
 - seq64::mainwnd, 191
- m_slots_chars
 - seq64::eventslots, 127
- m_slots_y
 - seq64::eventslots, 127
- m_smf0_channels
 - seq64::midi_splitter, 207
- m_smf0_channels_count
 - seq64::midi_splitter, 207
- m_smf0_seq_number
 - seq64::midi_splitter, 207
- m_smf0_splitter
 - seq64::midifile, 228
- m_snap_tick
 - seq64::sequence, 359
- m_starting_tick
 - seq64::perform, 270
- m_status
 - seq64::event, 101
- m_sysex
 - seq64::event, 101
- m_table
 - seq64::seqedit, 306
- m_text_x
 - seq64::user_settings, 389
- m_tick
 - seq64::maintime, 174
 - seq64::perform, 270
- m_time_beat_width
 - seq64::sequence, 359
- m_time_beats_per_measure
 - seq64::sequence, 359
- m_timestamp
 - seq64::event_list::event_key, 103
- m_top_index
 - seq64::eventslots, 127
- m_us_per_quarter_note
 - seq64::sequence, 359
- m_use_new_font

- seq64::user_settings, 388
- m_v_page_increment
 - seq64::perroll, 279
- m_v_perf_page_increment
 - seq64::user_settings, 389
- m_white_pixmap
 - seq64::font, 130
- m_window
 - seq64::gui_drawingarea_gtk2, 143
- m_window_redraw_rate_ms
 - seq64::user_settings, 389
- m_window_x
 - seq64::gui_drawingarea_gtk2, 143
 - seq64::gui_window_gtk2, 147
- m_x
 - seq64::click, 74
- m_xy_offset
 - seq64::eventslots, 127
 - seq64::perfnames, 243
- m_y
 - seq64::click, 74
- m_y_on_b_pixmap
 - seq64::font, 130
- m_zoom
 - seq64::seqedit, 305
- maintime
 - seq64::maintime, 171
- mainwid
 - seq64::mainwid, 177
- mainwid_border
 - seq64::user_settings, 385
- mainwid_grid_style_t
 - seq64::user_settings, 383
- mainwid_spacing
 - seq64::user_settings, 385
- mainwnd
 - seq64::mainwnd, 186
- mainwnd_cols
 - seq64::user_settings, 384
- mainwnd_key_event
 - seq64::perform, 265
- mainwnd_rows
 - seq64::user_settings, 384
- make_directory
 - seq64, 61
- mark_all
 - seq64::event_list, 107
- mark_out_of_range
 - seq64::event_list, 107
- mark_selected
 - seq64::sequence, 354
- mastermidibus
 - seq64::mastermidibus, 193
- max_sets
 - seq64::user_settings, 385
- mc_baseline_ppqn
 - seq64::user_settings, 392
- mc_max_zoom
 - seq64::user_settings, 391
- mc_min_zoom
 - seq64::user_settings, 391
- measures_to_ticks
 - seq64, 58
- measurestring_to_pulses
 - seq64, 51
- merge
 - seq64::event_list, 106
- midi_beat_width
 - seq64::user_settings, 386
- midi_beats_per_bar
 - seq64::user_settings, 386
- midi_beats_per_minute
 - seq64::user_settings, 386
- midi_buss_override
 - seq64::user_settings, 386
- midi_control_off
 - seq64::perform, 255
- midi_control_on
 - seq64::perform, 255
- midi_control_toggle
 - seq64::perform, 255
- midi_measures_to_pulses
 - seq64, 52
- midi_ppqn
 - seq64::user_settings, 386
- midi_splitter
 - seq64::midi_splitter, 205
- midibus
 - seq64::midibus, 212, 213
- midifile
 - seq64::midifile, 218
- midipulse
 - seq64, 48
- min
 - seq64, 65
- mod_last_tick
 - seq64::sequence, 343
- mod_timestamp
 - seq64::event, 99
- modify
 - seq64::perform, 252
- modify_current_event
 - seq64::eventslots, 122
- move
 - seq64::triggers, 368
- move_selected
 - seq64::triggers, 367
- move_selected_notes
 - seq64::sequence, 351
- move_selected_triggers_to
 - seq64::sequence, 348
- move_triggers
 - seq64::perform, 254
 - seq64::sequence, 349
- name_change_callback
 - seq64::seqedit, 303

- name_to_value
 - seq64::editable_event, [83](#)
- new_file
 - seq64::mainwnd, [189](#)
- new_sequence
 - seq64::perform, [252](#)
- next
 - seq64::triggers, [369](#)
- next_data_line
 - seq64::configfile, [77](#)
- next_trigger
 - seq64::triggers, [370](#)
- off_playing_notes
 - seq64::sequence, [355](#)
- off_queued
 - seq64::sequence, [343](#)
- on_button_press_event
 - seq64::Seq24PerfInput, [288](#)
 - seq64::Seq24SeqEventInput, [289](#)
 - seq64::Seq24SeqRollInput, [290](#)
 - seq64::mainwid, [180](#)
 - seq64::perfroll, [278](#)
 - seq64::seqdata, [295](#)
 - seq64::seqevent, [311](#)
 - seq64::seqkeys, [315](#)
 - seq64::seqtime, [332](#)
- on_button_release_event
 - seq64::Seq24PerfInput, [288](#)
 - seq64::Seq24SeqEventInput, [289](#)
 - seq64::Seq24SeqRollInput, [290](#)
 - seq64::mainwid, [181](#)
 - seq64::perfroll, [278](#)
 - seq64::seqdata, [295](#)
 - seq64::seqevent, [311](#)
 - seq64::seqkeys, [315](#)
 - seq64::seqtime, [332](#)
- on_delete_event
 - seq64::eventedit, [115](#)
 - seq64::mainwnd, [190](#)
 - seq64::seqedit, [305](#)
- on_enter_notify_event
 - seq64::seqkeys, [315](#)
- on_expose_event
 - seq64::eventslots, [126](#)
 - seq64::maintime, [173](#)
 - seq64::mainwid, [180](#)
 - seq64::perfnames, [242](#)
 - seq64::perfroll, [278](#)
 - seq64::perftime, [283](#)
 - seq64::seqdata, [295](#)
- on_focus_in_event
 - seq64::eventslots, [126](#)
 - seq64::mainwid, [181](#)
- on_focus_out_event
 - seq64::mainwid, [181](#)
- on_grouplearnchange
 - seq64::mainwnd, [190](#)
- on_key_press_event
 - seq64::eventedit, [114](#)
 - seq64::keybindentry, [159](#)
 - seq64::mainwnd, [190](#)
 - seq64::perfroll, [278](#)
 - seq64::seqedit, [305](#)
 - seq64::seqevent, [311](#)
 - seq64::seqroll, [328](#)
- on_key_release_event
 - seq64::mainwnd, [190](#)
- on_leave_notify_event
 - seq64::seqkeys, [316](#)
- on_motion_notify_event
 - seq64::Seq24PerfInput, [288](#)
 - seq64::Seq24SeqEventInput, [290](#)
 - seq64::mainwid, [181](#)
 - seq64::seqdata, [295](#)
 - seq64::seqevent, [311](#)
 - seq64::seqkeys, [315](#)
- on_move_down
 - seq64::eventslots, [126](#)
- on_move_up
 - seq64::eventslots, [126](#)
- on_realize
 - seq64::eventslots, [126](#)
 - seq64::gui_drawingarea_gtk2, [143](#)
 - seq64::maintime, [173](#)
 - seq64::mainwid, [180](#)
 - seq64::perfnames, [242](#)
 - seq64::perfroll, [278](#)
 - seq64::perftime, [283](#)
 - seq64::seqdata, [295](#)
 - seq64::seqevent, [311](#)
 - seq64::seqkeys, [315](#)
 - seq64::seqtime, [332](#)
- on_scroll_event
 - seq64::seqdata, [295](#)
 - seq64::seqedit, [305](#)
 - seq64::seqkeys, [316](#)
 - seq64::seqroll, [329](#)
- on_size_allocate
 - seq64::eventslots, [126](#)
 - seq64::perfnames, [242](#)
 - seq64::seqkeys, [316](#)
- open_file
 - seq64::mainwnd, [187](#)
- open_performance_edit
 - seq64::mainwnd, [189](#)
- open_performance_edit_2
 - seq64::mainwnd, [189](#)
- operator<
 - seq64::event, [96](#)
 - seq64::event_list::event_key, [102](#)
 - seq64::trigger, [361](#)
- operator=
 - seq64::click, [74](#)
 - seq64::editable_events, [89](#)
 - seq64::event, [96](#)
 - seq64::event_list, [105](#)

- seq64::keystroke, 167
- seq64::triggers, 364
- output
 - seq64::jack_assistant, 152
- output_func
 - seq64::perform, 261
- output_thread_func
 - seq64, 64
- page_movement
 - seq64::eventslots, 124
- page_topper
 - seq64::eventslots, 125
- parse
 - seq64::midifile, 218
 - seq64::optionsfile, 232
 - seq64::userfile, 393
- parse_command_line_options
 - seq64, 59
- parse_options_files
 - seq64, 59
- parse_prop_header
 - seq64::midifile, 220
- parse_proprietary_track
 - seq64::midifile, 221
- parse_smf_0
 - seq64::midifile, 219
- parse_smf_1
 - seq64::midifile, 220
- partial_assign
 - seq64::sequence, 341
- paste
 - seq64::triggers, 367
- paste_selected
 - seq64::sequence, 351
- paste_trigger
 - seq64::sequence, 348
- pause
 - seq64::sequence, 355
- pause_key
 - seq64::perform, 263
- pause_playing
 - seq64::mainwnd, 189
 - seq64::perfedit, 238
 - seq64::perform, 263
- perf_h_page_increment
 - seq64::user_settings, 385
- perf_v_page_increment
 - seq64::user_settings, 385
- perfedit
 - seq64::perfedit, 236
- perfnames
 - seq64::perfnames, 242
- perform
 - seq64::perform, 251
- perfroll_key_event
 - seq64::perform, 265
- perftime
 - seq64::perftime, 282
- play
 - seq64::mastermidibus, 197
 - seq64::midibus, 214
 - seq64::perform, 259
 - seq64::sequence, 345
 - seq64::triggers, 364
- play_note_off
 - seq64::sequence, 355
- play_note_on
 - seq64::sequence, 354
- playback_key_event
 - seq64::perform, 265
- poll_for_midi
 - seq64::mastermidibus, 196
- pop_redo
 - seq64::sequence, 341
- pop_undo
 - seq64::sequence, 341
- popup_event_menu
 - seq64::seqedit, 304
- popup_menu
 - seq64::seqmenu, 320
- popup_midibus_menu
 - seq64::seqedit, 304
- popup_sequence_menu
 - seq64::seqedit, 304
- popup_tool_menu
 - seq64::seqedit, 304
- port_exit
 - seq64::mastermidibus, 197
- port_start
 - seq64::mastermidibus, 197
- position
 - seq64::jack_assistant, 151
 - seq64::midi_container, 201
- position_jack
 - seq64::perform, 257
- pow2
 - seq64::midifile, 221
- ppqn
 - seq64::mainwnd, 187
 - seq64::midi_splitter, 206
 - seq64::midifile, 219
- ppqn_is_valid
 - seq64, 62
- print
 - seq64::sequence, 344
 - seq64::triggers, 364
- print_triggers
 - seq64::sequence, 344
- private_bus
 - seq64::user_settings, 386
- private_instrument
 - seq64::user_settings, 386
- process_events
 - seq64::lash, 169
- progress_color
 - seq64::gui_palette_gtk2, 145

- prop_item_size
 - seq64::midifile, 226
- pulse_length_us
 - seq64, 55
- pulses_to_measurestring
 - seq64, 50
- pulses_to_midi_measures
 - seq64, 50
- pulses_to_string
 - seq64, 50
- pulses_to_timestring
 - seq64, 51
- push_trigger_undo
 - seq64::perform, 254
 - seq64::sequence, 341
- push_undo
 - seq64::sequence, 341
- put
 - seq64::midi_container, 201
 - seq64::midi_list, 203
 - seq64::midi_vector, 210
- put_event_on_bus
 - seq64::sequence, 357
- read_byte_array
 - seq64::midifile, 223
- read_long
 - seq64::midifile, 222
- read_seq_number
 - seq64::midifile, 224
- read_track_name
 - seq64::midifile, 224
- read_varinum
 - seq64::midifile, 222
- redraw
 - seq64::mainwid, 177
 - seq64::seqevent, 309
 - seq64::seqroll, 327
- remove
 - seq64::event_list, 106
 - seq64::sequence, 358
 - seq64::triggers, 366
- remove_all
 - seq64::sequence, 358
- remove_marked
 - seq64::sequence, 354
- render_string
 - seq64::gui_drawingarea_gtk2, 139
- render_string_on_drawable
 - seq64::font, 129
- render_string_on_pixmap
 - seq64::gui_drawingarea_gtk2, 139
- reset
 - seq64::seqdata, 294
 - seq64::seqroll, 327
 - seq64::sequence, 355
- reset_draw_marker
 - seq64::sequence, 355
- reset_draw_trigger_marker
 - seq64::sequence, 355
- reset_sequences
 - seq64::perform, 259
- s_global_lash_driver
 - seq64, 69
- SEQ64_SUPER_MASK
 - seq64, 48
- save_events
 - seq64::editable_events, 90
 - seq64::eventslots, 122
- save_file
 - seq64::mainwnd, 190
- save_playing_state
 - seq64::perform, 262
- scroll_adjust
 - seq64::gui_drawingarea_gtk2, 142
 - seq64::gui_window_gtk2, 147
- select
 - seq64::triggers, 367
- select_action_e
 - seq64::sequence, 341
- select_all
 - seq64::sequence, 350
- select_event
 - seq64::eventslots, 123
- select_events
 - seq64::sequence, 350
- select_group_mute
 - seq64::perform, 256
- select_mute_group
 - seq64::perform, 256
- select_note_events
 - seq64::sequence, 349
- select_trigger
 - seq64::sequence, 346
- selected_trigger_end
 - seq64::sequence, 349
- selected_trigger_start
 - seq64::sequence, 348
- seq64, 41
 - adjustment_dummy, 65
 - beats_per_minute_from_tempo, 55
 - c_chord_text, 68
 - c_controller_names, 65
 - c_mainwid_x, 69
 - c_max_instruments, 68
 - c_midi_track_ctrl, 67
 - c_midibus, 66
 - c_midibus_output_size, 66
 - c_music_scales, 49
 - c_scales_policy, 67
 - c_scales_transpose_dn_neg, 68
 - c_scales_transpose_up, 67
 - c_select_all_notes, 69
 - choose_ppqn, 62
 - clock_e, 49
 - clock_tick_duration_bogus, 57
 - clock_ticks_from_ppqn, 57

create_lash_driver, 64
 DRAW_FIN, 49
 delete_lash_driver, 64
 delta_time_us_to_ticks, 56
 double_ticks_from_ppqn, 57
 draw_type, 49
 e_clock_mod, 49
 e_clock_off, 49
 e_clock_pos, 49
 EVENT_MIDI_SYSEX, 65
 EVENT_NOTE_OFF, 65
 EVENT_NULL_CHANNEL, 66
 extract_timing_numbers, 50
 file_accessible, 61
 file_executable, 61
 file_exists, 60
 file_is_directory, 61
 file_readable, 60
 file_writable, 60
 font_render, 65
 help_check, 58
 interaction_method_t, 49
 jack_session_callback, 63
 jack_shutdown_callback, 63
 jack_sync_callback, 62
 jack_timebase_callback, 63
 lash_driver, 64
 log2_time_sig_value, 54
 long_options, 69
 make_directory, 61
 measures_to_ticks, 58
 measurestring_to_pulses, 51
 midi_measures_to_pulses, 52
 midipulse, 48
 min, 65
 output_thread_func, 64
 parse_command_line_options, 59
 parse_options_files, 59
 ppqn_is_valid, 62
 pulse_length_us, 55
 pulses_to_measurestring, 50
 pulses_to_midi_measures, 50
 pulses_to_string, 50
 pulses_to_timestring, 51
 s_global_lash_driver, 69
 SEQ64_SUPER_MASK, 48
 seq_event_type_t, 48
 seq_modifier_t, 48
 seq_scroll_direction_t, 48
 shorten_file_spec, 53
 string_is_void, 54
 string_not_void, 53
 string_to_midibyte, 53
 string_to_pulses, 52
 strings_match, 54
 tempo_from_beats_per_minute, 55
 tempo_to_bytes, 55
 ticks_to_delta_time_us, 56
 timestring_to_pulses, 52
 to_string, 60
 versiontext, 68
 write_options_files, 60
 seq64::AbstractPerfInput, 71
 seq64::Seq24PerfInput, 287
 handle_motion_key, 288
 on_button_press_event, 288
 on_button_release_event, 288
 on_motion_notify_event, 288
 set_adding, 288
 seq64::Seq24SeqEventInput, 289
 on_button_press_event, 289
 on_button_release_event, 289
 on_motion_notify_event, 290
 set_adding, 289
 seq64::Seq24SeqRollInput, 290
 on_button_press_event, 290
 on_button_release_event, 290
 set_adding, 290
 seq64::automutex, 72
 seq64::click, 72
 click, 73, 74
 m_button, 74
 m_modifier, 74
 m_x, 74
 m_y, 74
 operator=, 74
 seq64::condition_var, 75
 cond, 76
 seq64::configfile, 76
 configfile, 77
 line_after, 78
 m_line, 78
 next_data_line, 77
 seq64::editable_event, 78
 analyze, 85
 category, 84
 category_channel_message, 82
 category_meta_event, 82
 category_name, 82
 category_prop_event, 82
 category_system_message, 82
 category_t, 82
 editable_event, 83
 format_timestamp, 85
 m_category, 87
 m_format_timestamp, 87
 m_name_meta, 87
 m_name_status, 87
 m_parent, 86
 name_to_value, 83
 set_status_from_string, 85
 sm_category_arrays, 86
 sm_category_names, 86
 sm_channel_event_names, 86
 sm_meta_event_names, 86
 sm_prop_event_names, 86

- sm_system_event_names, 86
- stock_event_string, 85
- time_as_measures, 84
- time_as_minutes, 84
- timestamp, 84
- timestamp_format_t, 82
- timestamp_measures, 82
- timestamp_pulses, 82
- timestamp_time, 82
- value_to_name, 83
- seq64::editable_events, 87
 - add, 90
 - count, 90
 - editable_events, 89
 - Key, 89
 - load_events, 89
 - m_current_event, 91
 - m_midi_parameters, 91
 - m_sequence, 91
 - operator=, 89
 - save_events, 90
- seq64::event, 91
 - ~event, 96
 - append_sysex, 100
 - check_channel, 97
 - event, 96
 - get_data, 100
 - get_rank, 101
 - is_channel_msg, 97
 - is_desired_cc_or_not_cc, 98
 - is_note_msg, 98
 - is_one_byte_msg, 98
 - is_two_byte_msg, 98
 - m_channel, 101
 - m_data, 101
 - m_has_link, 101
 - m_status, 101
 - m_sysex, 101
 - mod_timestamp, 99
 - operator<, 96
 - operator=, 96
 - set_channel, 100
 - set_data, 100
 - set_status, 99
- seq64::event_list, 103
 - add, 105
 - any_selected_notes, 108
 - clear, 106
 - count, 105
 - count_selected_events, 108
 - event_list, 105
 - link_new, 107
 - m_is_modified, 108
 - mark_all, 107
 - mark_out_of_range, 107
 - merge, 106
 - operator=, 105
 - remove, 106
 - unmodify, 106
 - verify_and_link, 107
- seq64::event_list::event_key, 101
 - event_key, 102
 - m_rank, 103
 - m_timestamp, 103
 - operator<, 102
- seq64::eventedit, 108
 - ~eventedit, 112
 - eventedit, 111
 - handle_insert, 114
 - handle_modify, 114
 - handle_save, 114
 - on_delete_event, 115
 - on_key_press_event, 114
 - set_dirty, 113
 - set_event_category, 113
 - set_event_data_0, 113
 - set_event_data_1, 113
 - set_event_name, 113
 - set_event_timestamp, 113
 - set_seq_ppqn, 112
 - set_seq_time_sig, 112
 - set_seq_title, 112
 - v_adjustment, 113, 114
- seq64::eventslots, 115
 - change_vert, 124
 - convert_y, 123
 - decrement_bottom, 125
 - decrement_current, 125
 - decrement_top, 125
 - delete_current_event, 121
 - draw_event, 123
 - draw_events, 124
 - increment_bottom, 126
 - increment_current, 125
 - increment_top, 125
 - insert_event, 120, 121
 - load_events, 120
 - m_char_w, 127
 - m_current_index, 127
 - m_setbox_w, 127
 - m_slots_chars, 127
 - m_slots_y, 127
 - m_top_index, 127
 - m_xy_offset, 127
 - modify_current_event, 122
 - on_expose_event, 126
 - on_focus_in_event, 126
 - on_move_down, 126
 - on_move_up, 126
 - on_realize, 126
 - on_size_allocate, 126
 - page_movement, 124
 - page_topper, 125
 - save_events, 122
 - select_event, 123
 - set_current_event, 120

- set_text, 123
- seq64::font, 127
 - BLACK_ON_CYAN, 129
 - BLACK_ON_YELLOW, 129
 - BLACK, 129
 - CYAN_ON_BLACK, 129
 - Color, 129
 - init, 129
 - m_b_on_c_pixmap, 130
 - m_b_on_y_pixmap, 130
 - m_black_pixmap, 130
 - m_c_on_b_pixmap, 130
 - m_clip_mask, 130
 - m_font_h, 130
 - m_font_w, 129
 - m_pixmap, 130
 - m_white_pixmap, 130
 - m_y_on_b_pixmap, 130
 - render_string_on_drawable, 129
 - WHITE, 129
 - YELLOW_ON_BLACK, 129
- seq64::gui_assistant, 131
 - gui_assistant, 132
- seq64::gui_assistant_gtk2, 132
 - jack_idle_connect, 133
 - lash_timeout_connect, 133
 - sm_internal_keys, 134
- seq64::gui_drawingarea_gtk2, 134
 - clear_window, 137
 - draw_line, 137–139
 - draw_line_on_pixmap, 138
 - draw_normal_rectangle_on_pixmap, 142
 - draw_rectangle, 140, 141
 - draw_rectangle_on_pixmap, 142
 - m_background, 143
 - m_drop_x, 143
 - m_foreground, 143
 - m_mainperf, 143
 - m_pixmap, 143
 - m_window, 143
 - m_window_x, 143
 - on_realize, 143
 - render_string, 139
 - render_string_on_pixmap, 139
 - scroll_adjust, 142
 - set_line, 137
- seq64::gui_drawingarea_gtk2::rect, 286
- seq64::gui_palette_gtk2, 144
 - gui_palette_gtk2, 145
 - line_color, 145
 - progress_color, 145
- seq64::gui_window_gtk2, 146
 - gui_window_gtk2, 147
 - m_redraw_period_ms, 147
 - m_window_x, 147
 - scroll_adjust, 147
- seq64::jack_assistant, 148
 - ~jack_assistant, 150
 - client_open, 153
 - deinit, 150
 - error_message, 153
 - info_message, 152
 - init, 150
 - jack_assistant, 150
 - jack_session_callback, 157
 - jack_shutdown_callback, 155
 - jack_sync_callback, 156
 - jack_timebase_callback, 156
 - output, 152
 - position, 151
 - session_event, 151
 - set_beats_per_minute, 150
 - set_position, 155
 - set_ppqn, 152
 - show_position, 154
 - show_statuses, 153
 - sm_status_pairs, 157
 - start, 151
 - stop, 151
 - sync, 155
- seq64::jack_scratchpad, 157
- seq64::keybindentry, 157
 - events, 158
 - groups, 158
 - keybindentry, 158
 - location, 158
 - m_key, 159
 - on_key_press_event, 159
 - set, 159
 - type, 158
- seq64::keys_perform, 159
 - ~keys_perform, 161
 - get_keys, 162
 - key_name, 162
 - m_key_bpm_up, 163
 - m_key_show_ui_sequence_number, 163
 - set_all_key_events, 162
 - set_all_key_groups, 162
 - set_key_event, 162
 - set_key_group, 163
 - set_keys, 161
 - show_ui_sequence_key, 162
 - show_ui_sequence_number, 162
- seq64::keys_perform_gtk2, 163
 - ~keys_perform_gtk2, 165
 - key_name, 165
 - set_all_key_events, 165
 - set_all_key_groups, 165
- seq64::keys_perform_transfer, 165
- seq64::keystroke, 166
 - is_letter, 167
 - keystroke, 166, 167
 - m_is_press, 167
 - m_key, 167
 - m_modifier, 168
 - operator=, 167

- seq64::lash, 168
 - handle_config, 169
 - handle_event, 169
 - init, 169
 - lash, 168
 - process_events, 169
 - set_alsa_client_id, 169
- seq64::maintime, 169
 - idle_progress, 172
 - m_bar_width, 173
 - m_beat_width, 173
 - m_box_height, 173
 - m_box_width, 173
 - m_flash_height, 173
 - m_flash_width, 173
 - m_ppqn, 174
 - m_tick, 174
 - maintime, 171
 - on_expose_event, 173
 - on_realize, 173
- seq64::mainwid, 174
 - calculate_base_sizes, 180
 - draw_marker_on_sequence, 178
 - draw_sequence_on_pixmap, 178
 - draw_sequence_pixmap_on_window, 179
 - draw_sequences_on_pixmap, 179
 - m_button_down, 182
 - m_moving_seq, 182
 - m_screenset_offset, 182
 - m_screenset_slots, 182
 - mainwid, 177
 - on_button_press_event, 180
 - on_button_release_event, 181
 - on_expose_event, 180
 - on_focus_in_event, 181
 - on_focus_out_event, 181
 - on_motion_notify_event, 181
 - on_realize, 180
 - redraw, 177
 - seq_from_xy, 179
 - set_screenset, 177
 - timeout, 179
 - update_markers, 178
 - update_sequences_on_window, 178
 - valid_sequence, 178
- seq64::mainwnd, 182
 - about_dialog, 187
 - adj_callback_bpm, 188
 - adj_callback_ss, 188
 - edit_callback_notepad, 188
 - file_import_dialog, 187
 - m_adjust_load_offset, 191
 - m_button_play, 191
 - m_main_wid, 191
 - m_ppqn, 191
 - m_sigpipe, 191
 - mainwnd, 186
 - new_file, 189
 - on_delete_event, 190
 - on_grouplearnchange, 190
 - on_key_press_event, 190
 - on_key_release_event, 190
 - open_file, 187
 - open_performance_edit, 189
 - open_performance_edit_2, 189
 - pause_playing, 189
 - ppqn, 187
 - save_file, 190
 - set_image, 188
 - signal_action, 190
 - start_playing, 188
 - stop_playing, 189
 - timer_callback, 188
 - toggle_playing, 189
 - update_window_title, 189
- seq64::mastermidibus, 191
 - ~mastermidibus, 194
 - clock, 195
 - continue_from, 195
 - flush, 195
 - get_clock, 198
 - get_input, 198
 - get_midi_event, 196
 - get_midi_in_bus_name, 195
 - get_midi_out_bus_name, 194
 - init, 194
 - init_clock, 196
 - is_more_input, 196
 - m_beats_per_minute, 198
 - mastermidibus, 193
 - play, 197
 - poll_for_midi, 196
 - port_exit, 197
 - port_start, 197
 - set_beats_per_minute, 194
 - set_clock, 197
 - set_input, 198
 - set_ppqn, 194
 - set_sequence_input, 196
 - start, 195
 - stop, 195
 - sysex, 197
- seq64::midi_container, 199
 - add_long, 201
 - add_variable, 201
 - fill, 200
 - get, 201
 - position, 201
 - put, 201
- seq64::midi_list, 202
 - CharList, 203
 - get, 203
 - put, 203
- seq64::midi_measures, 203
 - m_divisions, 204
- seq64::midi_splitter, 204

- increment, 206
- m_smf0_channels, 207
- m_smf0_channels_count, 207
- m_smf0_seq_number, 207
- midi_splitter, 205
- ppqn, 206
- split, 206
- split_channel, 206
- seq64::midi_timing, 207
 - m_beat_width, 209
 - m_beats_per_measure, 208
 - m_beats_per_minute, 208
 - m_ppqn, 209
- seq64::midi_vector, 209
 - get, 210
 - put, 210
- seq64::midibus, 210
 - clock, 214
 - continue_from, 214
 - deinit_in, 213
 - init_clock, 214
 - init_in, 213
 - init_in_sub, 214
 - init_out, 213
 - init_out_sub, 213
 - m_clock_mod, 215
 - midibus, 212, 213
 - play, 214
 - set_input, 215
 - sysex, 214
- seq64::midifile, 215
 - add_trigger, 222
 - checklen, 222
 - errdump, 226
 - is_sysex_special_id, 226
 - m_char_list, 227
 - m_data, 227
 - m_disable_reported, 227
 - m_error_is_fatal, 227
 - m_error_message, 227
 - m_file_size, 227
 - m_new_format, 227
 - m_pos, 227
 - m_smf0_splitter, 228
 - midifile, 218
 - parse, 218
 - parse_prop_header, 220
 - parse_proprietary_track, 221
 - parse_smf_0, 219
 - parse_smf_1, 220
 - pow2, 221
 - ppqn, 219
 - prop_item_size, 226
 - read_byte_array, 223
 - read_long, 222
 - read_seq_number, 224
 - read_track_name, 224
 - read_varinum, 222
 - varinum_size, 225
 - write, 219
 - write_byte, 223
 - write_long, 222
 - write_prop_header, 224
 - write_proprietary_track, 225
 - write_seq_number, 224
 - write_short, 223
 - write_track_name, 223
 - write_varinum, 223
- seq64::mutex, 228
 - sm_recursive_mutex, 229
- seq64::options, 229
 - button, 230
 - e_jack_connect, 230
 - e_jack_disconnect, 230
 - e_jack_master, 230
 - e_jack_master_cond, 230
 - e_jack_start_mode_live, 230
 - e_jack_start_mode_song, 230
 - e_jack_transport, 230
 - m_notebook, 230
- seq64::optionsfile, 230
 - parse, 232
 - write, 233
- seq64::perfedit, 233
 - ~perfedit, 237
 - collapse, 238
 - copy, 238
 - enqueue_draw, 237
 - enregister_peer, 237
 - expand, 238
 - grow, 238
 - init_before_show, 237
 - m_bpm, 239
 - m_bw, 239
 - pause_playing, 238
 - perfedit, 236
 - set_beat_width, 237
 - set_beats_per_bar, 237
 - set_guides, 237
 - start_playing, 238
 - timeout, 238
 - toggle_playing, 238
 - undo, 238
 - zoom_check, 237
- seq64::perfnames, 239
 - draw_sequence, 242
 - enqueue_draw, 242
 - m_char_w, 243
 - m_namebox_w, 243
 - m_names_chars, 242
 - m_names_y, 243
 - m_parent, 242
 - m_setbox_w, 243
 - m_xy_offset, 243
 - on_expose_event, 242
 - on_realize, 242

- on_size_allocate, 242
- perfnames, 242
- seq64::perform, 243
 - ~perform, 251
 - add_sequence, 253
 - all_notes_off, 257
 - clamp_track, 268
 - clear_all, 252
 - clear_sequence_triggers, 253
 - copy_triggers, 254
 - decrement_beats_per_minute, 263
 - deinit_jack, 266
 - delete_sequence, 253
 - finish, 253
 - get_group_mute_state, 261
 - get_max_tick, 254
 - get_max_trigger, 255
 - get_screen_set_notepad, 255
 - get_sequence, 259
 - handle_midi_control, 255
 - highlight, 264
 - increment_beats_per_minute, 264
 - init_jack, 266
 - inner_start, 268
 - inner_stop, 268
 - install_sequence, 267
 - is_active, 259
 - is_dirty_edit, 258
 - is_dirty_main, 258
 - is_dirty_names, 258
 - is_dirty_perf, 258
 - is_midi_control_valid, 266
 - is_mseq_valid, 267
 - is_screenset_valid, 266
 - is_seq_valid, 267
 - jack_sync_callback, 269
 - launch, 252
 - launch_input_thread, 265
 - launch_output_thread, 265
 - lookup_keyevent_key, 262
 - m_beat_width, 270
 - m_beats_per_bar, 270
 - m_is_modified, 271
 - m_left_tick, 270
 - m_max_sets, 270
 - m_one_measure, 270
 - m_playback_mode, 269
 - m_playscreen_offset, 269
 - m_right_tick, 270
 - m_seqs, 269
 - m_seqs_in_set, 270
 - m_sequence_count, 271
 - m_sequence_max, 271
 - m_starting_tick, 270
 - m_tick, 270
 - mainwnd_key_event, 265
 - midi_control_off, 255
 - midi_control_on, 255
 - midi_control_toggle, 255
 - modify, 252
 - move_triggers, 254
 - new_sequence, 252
 - output_func, 261
 - pause_key, 263
 - pause_playing, 263
 - perform, 251
 - perfroll_key_event, 265
 - play, 259
 - playback_key_event, 265
 - position_jack, 257
 - push_trigger_undo, 254
 - reset_sequences, 259
 - save_playing_state, 262
 - select_group_mute, 256
 - select_mute_group, 256
 - seq_in_playing_screen, 266
 - sequence_count, 252
 - sequence_label, 264
 - sequence_playing_off, 260
 - sequence_playing_on, 260
 - set_active, 257
 - set_beats_per_minute, 260
 - set_group_mute_state, 260
 - set_input_bus, 264
 - set_key_event, 268
 - set_key_group, 268
 - set_left_tick, 253
 - set_offset, 261
 - set_orig_ticks, 260
 - set_playing_screenset, 256
 - set_right_tick, 254
 - set_screen_set_notepad, 256
 - set_screenset, 256
 - set_sequence_control_status, 260
 - set_was_active, 258
 - show_ui_sequence_key, 262
 - show_ui_sequence_number, 262
 - sm_mc_dummy, 269
 - start, 257
 - start_key, 263
 - start_playing, 262
 - stop, 257
 - stop_key, 263
 - stop_playing, 263
 - unset_mode_group_learn, 256
 - unset_sequence_control_status, 260
- seq64::performcallback, 271
- seq64::perfroll, 272
 - ~perfroll, 275
 - convert_x, 277
 - convert_xy, 277
 - draw_drawable_row, 278
 - draw_progress, 276
 - draw_sequence_on, 277
 - enqueue_draw, 278
 - fill_background_pixmap, 276

- FruityPerfInput, 279
- init_before_show, 276
- m_fruity_interaction, 279
- m_h_page_increment, 279
- m_parent, 279
- m_v_page_increment, 279
- on_button_press_event, 278
- on_button_release_event, 278
- on_expose_event, 278
- on_key_press_event, 278
- on_realize, 278
- set_guides, 276
- set_ppqn, 277
- set_zoom, 278
- snap_x, 277
- update_sizes, 276
- seq64::perftime, 280
 - enqueue_draw, 283
 - key_press_event, 283
 - m_left_marker_tick, 283
 - m_measure_length, 283
 - m_parent, 283
 - m_right_marker_tick, 284
 - on_expose_event, 283
 - on_realize, 283
 - perftime, 282
 - set_guides, 282
- seq64::rc_settings, 284
 - home_config_directory, 286
- seq64::rect, 286
- seq64::seqdata, 291
 - idle_redraw, 294
 - m_number_h, 296
 - m_number_offset_y, 296
 - m_number_w, 296
 - on_button_press_event, 295
 - on_button_release_event, 295
 - on_expose_event, 295
 - on_motion_notify_event, 295
 - on_realize, 295
 - on_scroll_event, 295
 - reset, 294
 - seqdata, 294
 - set_data_type, 294
 - set_zoom, 294
 - update_sizes, 294
 - xy_to_rect, 295
- seq64::seqedit, 296
 - apply_length, 302
 - create_menus, 303
 - do_action, 304
 - fill_top_bar, 303
 - get_measures, 302
 - horizontal_adjust, 301
 - m_initial_snap, 305
 - m_pos, 306
 - m_seqdata_wid, 306
 - m_seqkeys_wid, 306
 - m_seqtime_wid, 306
 - m_table, 306
 - m_zoom, 305
 - name_change_callback, 303
 - on_delete_event, 305
 - on_key_press_event, 305
 - on_scroll_event, 305
 - popup_event_menu, 304
 - popup_midibus_menu, 304
 - popup_sequence_menu, 304
 - popup_tool_menu, 304
 - seqedit, 301
 - set_background_sequence, 303
 - set_data_type, 303
 - set_key, 302
 - set_measures, 302
 - set_midi_bus, 302
 - set_midi_channel, 302
 - set_note_length, 301
 - set_scale, 302
 - set_snap, 301
 - set_zoom, 301
 - timeout, 304
- seq64::seqevent, 306
 - change_horz, 310
 - convert_t, 310
 - convert_x, 310
 - draw_background, 309
 - draw_events_on, 310
 - draw_pixmap_on_window, 310
 - drop_event, 310
 - idle_redraw, 310
 - on_button_press_event, 311
 - on_button_release_event, 311
 - on_key_press_event, 311
 - on_motion_notify_event, 311
 - on_realize, 311
 - redraw, 309
 - set_data_type, 309
 - set_snap, 309
 - snap_x, 311
 - start_paste, 310
 - update_sizes, 309
 - x_to_w, 310
- seq64::seqkeys, 312
 - draw_key, 315
 - on_button_press_event, 315
 - on_button_release_event, 315
 - on_enter_notify_event, 315
 - on_leave_notify_event, 316
 - on_motion_notify_event, 315
 - on_realize, 315
 - on_scroll_event, 316
 - on_size_allocate, 316
 - set_hint_state, 314
- seq64::seqmenu, 316
 - ~seqmenu, 319
 - is_edit_sequence, 320

- m_modified, [321](#)
- m_seqedit, [321](#)
- popup_menu, [320](#)
- seq_clear_perf, [321](#)
- seq_copy, [320](#)
- seq_cut, [320](#)
- seq_edit, [320](#)
- seq_event_edit, [320](#)
- seq_new, [320](#)
- seq_paste, [321](#)
- seqmenu, [319](#)
- seq64::seqroll, [321](#)
 - ~seqroll, [326](#)
 - change_horz, [328](#)
 - convert_tn, [328](#)
 - draw_events_on, [328](#)
 - draw_events_on_pixmap, [327](#)
 - draw_progress_on_window, [327](#)
 - follow_progress, [327](#)
 - horizontal_adjust, [327](#)
 - m_pos, [329](#)
 - on_key_press_event, [328](#)
 - on_scroll_event, [329](#)
 - redraw, [327](#)
 - reset, [327](#)
 - seqroll, [325](#)
 - set_background_sequence, [326](#)
 - set_data_type, [326](#)
 - set_key, [326](#)
 - set_scale, [326](#)
 - set_zoom, [326](#)
 - snap_x, [328](#)
 - snap_y, [328](#)
 - update_background, [327](#)
 - update_sizes, [326](#)
 - xy_to_rect, [328](#)
- seq64::seqtime, [329](#)
 - on_button_press_event, [332](#)
 - on_button_release_event, [332](#)
 - on_realize, [332](#)
 - seqtime, [331](#)
 - update_pixmap, [332](#)
- seq64::sequence, [332](#)
 - add_event, [345](#), [351](#)
 - add_note, [351](#)
 - add_trigger, [345](#)
 - adjust_trigger_offsets_to_length, [358](#)
 - background_sequence, [356](#)
 - change_event_data_range, [352](#)
 - clear_triggers, [349](#)
 - copy_events, [357](#)
 - copy_selected, [350](#)
 - copy_triggers, [349](#)
 - cut_selected, [351](#)
 - decrement_selected, [353](#)
 - del_trigger, [346](#)
 - e_deselect, [341](#)
 - e_is_selected, [341](#)
 - e_remove_one, [341](#)
 - e_select, [341](#)
 - e_select_one, [341](#)
 - e_toggle_selection, [341](#)
 - e_would_select, [341](#)
 - event_count, [341](#)
 - fill_container, [356](#)
 - get_beat_width, [342](#)
 - get_clipboard_box, [351](#)
 - get_last_tick, [342](#)
 - get_max_trigger, [349](#)
 - get_minmax_note_events, [355](#)
 - get_name, [342](#)
 - get_next_event, [356](#)
 - get_next_note_event, [355](#)
 - get_num_selected_events, [350](#)
 - get_num_selected_notes, [350](#)
 - get_selected_box, [351](#)
 - get_trigger_state, [346](#)
 - grow_selected, [353](#)
 - grow_trigger, [346](#)
 - increment_selected, [353](#)
 - intersect_events, [347](#)
 - intersect_notes, [347](#)
 - intersect_triggers, [347](#)
 - is_dirty_edit, [344](#)
 - is_dirty_main, [344](#)
 - is_dirty_names, [344](#)
 - is_dirty_perf, [344](#)
 - link_new, [354](#)
 - m_32nds_per_quarter, [359](#)
 - m_background_sequence, [360](#)
 - m_clocks_per_metronome, [359](#)
 - m_length, [359](#)
 - m_midi_channel, [358](#)
 - m_musical_key, [360](#)
 - m_musical_scale, [360](#)
 - m_mutex, [360](#)
 - m_parent, [358](#)
 - m_playing_notes, [359](#)
 - m_raise, [359](#)
 - m_seq_number, [359](#)
 - m_snap_tick, [359](#)
 - m_time_beat_width, [359](#)
 - m_time_beats_per_measure, [359](#)
 - m_us_per_quarter_note, [359](#)
 - mark_selected, [354](#)
 - mod_last_tick, [343](#)
 - move_selected_notes, [351](#)
 - move_selected_triggers_to, [348](#)
 - move_triggers, [349](#)
 - off_playing_notes, [355](#)
 - off_queued, [343](#)
 - partial_assign, [341](#)
 - paste_selected, [351](#)
 - paste_trigger, [348](#)
 - pause, [355](#)
 - play, [345](#)

- play_note_off, 355
- play_note_on, 354
- pop_redo, 341
- pop_undo, 341
- print, 344
- print_triggers, 344
- push_trigger_undo, 341
- push_undo, 341
- put_event_on_bus, 357
- remove, 358
- remove_all, 358
- remove_marked, 354
- reset, 355
- reset_draw_marker, 355
- reset_draw_trigger_marker, 355
- select_action_e, 341
- select_all, 350
- select_events, 350
- select_note_events, 349
- select_trigger, 346
- selected_trigger_end, 349
- selected_trigger_start, 348
- set_beat_width, 342
- set_beats_per_bar, 341
- set_dirty, 344
- set_dirty_mp, 344
- set_last_tick, 343
- set_length, 342
- set_master_midi_bus, 349
- set_midi_bus, 349
- set_midi_channel, 344
- set_parent, 357
- set_playing, 343
- set_quantized_rec, 343
- set_rec_vol, 342
- set_recording, 343
- set_snap_tick, 343
- set_thru, 344
- set_trigger_offset, 357
- split_trigger, 346, 357
- stream_event, 352
- stretch_selected, 353
- toggle_queued, 343
- transpose_notes, 356
- unpaint_all, 354
- unselect, 354
- unselect_triggers, 347
- verify_and_link, 354
- zero_markers, 354
- seq64::trigger, 360
 - operator<, 361
- seq64::triggers, 361
 - add, 365
 - adjust_offset, 370
 - adjust_offsets_to_length, 365
 - copy, 369
 - get_maximum, 368
 - get_selected_end, 368
 - get_selected_start, 368
 - get_state, 366
 - grow, 366
 - intersect, 367
 - m_length, 370
 - m_ppqn, 370
 - move, 368
 - move_selected, 367
 - next, 369
 - next_trigger, 370
 - operator=, 364
 - paste, 367
 - play, 364
 - print, 364
 - remove, 366
 - select, 367
 - set_length, 364
 - split, 365, 366
 - triggers, 364
 - unselect, 367
- seq64::user_instrument, 370
 - controller_active, 372
 - controller_max, 372
 - controller_name, 372
 - copy_definitions, 372
 - m_controller_count, 373
 - m_is_valid, 373
 - set_controller, 372
 - set_defaults, 372
 - set_name, 372
- seq64::user_instrument_t, 373
 - controllers, 373
 - controllers_active, 373
 - instrument, 373
- seq64::user_midi_bus, 374
 - channel_count, 375
 - channel_max, 375
 - copy_definitions, 375
 - instrument, 375
 - m_channel_count, 376
 - m_is_valid, 376
 - set_defaults, 375
 - set_instrument, 375
- seq64::user_midi_bus_t, 376
 - alias, 376
 - instrument, 376
- seq64::user_settings, 376
 - bus, 384
 - Busses, 383
 - control_height, 385
 - controller_active, 384
 - controller_name, 384
 - dump_summary, 385
 - get_globals, 384
 - grid_style_black, 383
 - grid_style_max, 383
 - grid_style_normal, 383
 - grid_style_white, 383

- instrument, [384](#)
- Instruments, [383](#)
- m_allow_two_perfedits, [388](#)
- m_control_height, [388](#)
- m_global_seq_feature_save, [388](#)
- m_gmute_tracks, [390](#)
- m_grid_brackets, [387](#)
- m_grid_style, [387](#)
- m_h_perf_page_increment, [389](#)
- m_instruments, [386](#)
- m_mainwid_border, [387](#)
- m_mainwid_x, [391](#)
- m_mainwnd_cols, [387](#)
- m_mainwnd_rows, [387](#)
- m_max_sequence, [391](#)
- m_max_sets, [387](#)
- m_midi_beat_width, [390](#)
- m_midi_beats_per_measure, [390](#)
- m_midi_beats_per_minute, [390](#)
- m_midi_buses, [386](#)
- m_midi_buss_override, [390](#)
- m_midi_ppqn, [390](#)
- m_progress_bar_colored, [389](#)
- m_progress_bar_thick, [389](#)
- m_save_user_config, [391](#)
- m_seqarea_seq_x, [391](#)
- m_seqarea_x, [391](#)
- m_seqchars_x, [389](#)
- m_seqedit_bgsequence, [388](#)
- m_seqedit_key, [388](#)
- m_seqedit_scale, [388](#)
- m_seqs_in_set, [390](#)
- m_text_x, [389](#)
- m_use_new_font, [388](#)
- m_v_perf_page_increment, [389](#)
- m_window_redraw_rate_ms, [389](#)
- mainwid_border, [385](#)
- mainwid_grid_style_t, [383](#)
- mainwid_spacing, [385](#)
- mainwnd_cols, [384](#)
- mainwnd_rows, [384](#)
- max_sets, [385](#)
- mc_baseline_ppqn, [392](#)
- mc_max_zoom, [391](#)
- mc_min_zoom, [391](#)
- midi_beat_width, [386](#)
- midi_beats_per_bar, [386](#)
- midi_beats_per_minute, [386](#)
- midi_buss_override, [386](#)
- midi_ppqn, [386](#)
- perf_h_page_increment, [385](#)
- perf_v_page_increment, [385](#)
- private_bus, [386](#)
- private_instrument, [386](#)
- set_defaults, [384](#)
- set_globals, [384](#)
- text_x, [385](#)
- text_y, [385](#)
- zoom, [384](#)
- seq64::userfile, [392](#)
 - dump_setting_summary, [393](#)
 - parse, [393](#)
 - write, [393](#)
- seq_clear_perf
 - seq64::seqmenu, [321](#)
- seq_copy
 - seq64::seqmenu, [320](#)
- seq_cut
 - seq64::seqmenu, [320](#)
- seq_edit
 - seq64::seqmenu, [320](#)
- seq_event_edit
 - seq64::seqmenu, [320](#)
- seq_event_type_t
 - seq64, [48](#)
- seq_from_xy
 - seq64::mainwid, [179](#)
- seq_in_playing_screen
 - seq64::perform, [266](#)
- seq_modifier_t
 - seq64, [48](#)
- seq_new
 - seq64::seqmenu, [320](#)
- seq_paste
 - seq64::seqmenu, [321](#)
- seq_scroll_direction_t
 - seq64, [48](#)
- seqdata
 - seq64::seqdata, [294](#)
- seqedit
 - seq64::seqedit, [301](#)
- seqmenu
 - seq64::seqmenu, [319](#)
- seqroll
 - seq64::seqroll, [325](#)
- seqtime
 - seq64::seqtime, [331](#)
- sequence_count
 - seq64::perform, [252](#)
- sequence_label
 - seq64::perform, [264](#)
- sequence_playing_off
 - seq64::perform, [260](#)
- sequence_playing_on
 - seq64::perform, [260](#)
- session_event
 - seq64::jack_assistant, [151](#)
- set
 - seq64::keybindentry, [159](#)
- set_active
 - seq64::perform, [257](#)
- set_adding
 - seq64::Seq24PerfInput, [288](#)
 - seq64::Seq24SeqEventInput, [289](#)
 - seq64::Seq24SeqRollInput, [290](#)
- set_all_key_events

- seq64::keys_perform, 162
- seq64::keys_perform_gtk2, 165
- set_all_key_groups
 - seq64::keys_perform, 162
 - seq64::keys_perform_gtk2, 165
- set_alsa_client_id
 - seq64::lash, 169
- set_background_sequence
 - seq64::seqedit, 303
 - seq64::seqroll, 326
- set_beat_width
 - seq64::perfedit, 237
 - seq64::sequence, 342
- set_beats_per_bar
 - seq64::perfedit, 237
 - seq64::sequence, 341
- set_beats_per_minute
 - seq64::jack_assistant, 150
 - seq64::mastermidibus, 194
 - seq64::perform, 260
- set_channel
 - seq64::event, 100
- set_clock
 - seq64::mastermidibus, 197
- set_controller
 - seq64::user_instrument, 372
- set_current_event
 - seq64::eventslots, 120
- set_data
 - seq64::event, 100
- set_data_type
 - seq64::seqdata, 294
 - seq64::seqedit, 303
 - seq64::seqevent, 309
 - seq64::seqroll, 326
- set_defaults
 - seq64::user_instrument, 372
 - seq64::user_midi_bus, 375
 - seq64::user_settings, 384
- set_dirty
 - seq64::eventedit, 113
 - seq64::sequence, 344
- set_dirty_mp
 - seq64::sequence, 344
- set_event_category
 - seq64::eventedit, 113
- set_event_data_0
 - seq64::eventedit, 113
- set_event_data_1
 - seq64::eventedit, 113
- set_event_name
 - seq64::eventedit, 113
- set_event_timestamp
 - seq64::eventedit, 113
- set_globals
 - seq64::user_settings, 384
- set_group_mute_state
 - seq64::perform, 260
- set_guides
 - seq64::perfedit, 237
 - seq64::perfroll, 276
 - seq64::perftime, 282
- set_hint_state
 - seq64::seqkeys, 314
- set_image
 - seq64::mainwnd, 188
- set_input
 - seq64::mastermidibus, 198
 - seq64::midibus, 215
- set_input_bus
 - seq64::perform, 264
- set_instrument
 - seq64::user_midi_bus, 375
- set_key
 - seq64::seqedit, 302
 - seq64::seqroll, 326
- set_key_event
 - seq64::keys_perform, 162
 - seq64::perform, 268
- set_key_group
 - seq64::keys_perform, 163
 - seq64::perform, 268
- set_keys
 - seq64::keys_perform, 161
- set_last_tick
 - seq64::sequence, 343
- set_left_tick
 - seq64::perform, 253
- set_length
 - seq64::sequence, 342
 - seq64::triggers, 364
- set_line
 - seq64::gui_drawingarea_gtk2, 137
- set_master_midi_bus
 - seq64::sequence, 349
- set_measures
 - seq64::seqedit, 302
- set_midi_bus
 - seq64::seqedit, 302
 - seq64::sequence, 349
- set_midi_channel
 - seq64::seqedit, 302
 - seq64::sequence, 344
- set_name
 - seq64::user_instrument, 372
- set_note_length
 - seq64::seqedit, 301
- set_offset
 - seq64::perform, 261
- set_orig_ticks
 - seq64::perform, 260
- set_parent
 - seq64::sequence, 357
- set_playing
 - seq64::sequence, 343
- set_playing_screensets
 -

- seq64::perform, 256
- set_position
 - seq64::jack_assistant, 155
- set_ppqn
 - seq64::jack_assistant, 152
 - seq64::mastermidibus, 194
 - seq64::perfroll, 277
- set_quantized_rec
 - seq64::sequence, 343
- set_rec_vol
 - seq64::sequence, 342
- set_recording
 - seq64::sequence, 343
- set_right_tick
 - seq64::perform, 254
- set_scale
 - seq64::seqedit, 302
 - seq64::seqroll, 326
- set_screen_set_notepad
 - seq64::perform, 256
- set_screenshot
 - seq64::mainwid, 177
 - seq64::perform, 256
- set_seq_ppqn
 - seq64::eventedit, 112
- set_seq_time_sig
 - seq64::eventedit, 112
- set_seq_title
 - seq64::eventedit, 112
- set_sequence_control_status
 - seq64::perform, 260
- set_sequence_input
 - seq64::mastermidibus, 196
- set_snap
 - seq64::seqedit, 301
 - seq64::seqevent, 309
- set_snap_tick
 - seq64::sequence, 343
- set_status
 - seq64::event, 99
- set_status_from_string
 - seq64::editable_event, 85
- set_text
 - seq64::eventslots, 123
- set_thru
 - seq64::sequence, 344
- set_trigger_offset
 - seq64::sequence, 357
- set_was_active
 - seq64::perform, 258
- set_zoom
 - seq64::perfroll, 278
 - seq64::seqdata, 294
 - seq64::seqedit, 301
 - seq64::seqroll, 326
- shorten_file_spec
 - seq64, 53
- show_position
 - seq64::jack_assistant, 154
- show_statuses
 - seq64::jack_assistant, 153
- show_ui_sequence_key
 - seq64::keys_perform, 162
 - seq64::perform, 262
- show_ui_sequence_number
 - seq64::keys_perform, 162
 - seq64::perform, 262
- signal_action
 - seq64::mainwnd, 190
- sm_category_arrays
 - seq64::editable_event, 86
- sm_category_names
 - seq64::editable_event, 86
- sm_channel_event_names
 - seq64::editable_event, 86
- sm_internal_keys
 - seq64::gui_assistant_gtk2, 134
- sm_mc_dummy
 - seq64::perform, 269
- sm_meta_event_names
 - seq64::editable_event, 86
- sm_prop_event_names
 - seq64::editable_event, 86
- sm_recursive_mutex
 - seq64::mutex, 229
- sm_status_pairs
 - seq64::jack_assistant, 157
- sm_system_event_names
 - seq64::editable_event, 86
- snap_x
 - seq64::perfroll, 277
 - seq64::seqevent, 311
 - seq64::seqroll, 328
- snap_y
 - seq64::seqroll, 328
- split
 - seq64::midi_splitter, 206
 - seq64::triggers, 365, 366
- split_channel
 - seq64::midi_splitter, 206
- split_trigger
 - seq64::sequence, 346, 357
- start
 - seq64::jack_assistant, 151
 - seq64::mastermidibus, 195
 - seq64::perform, 257
- start_key
 - seq64::perform, 263
- start_paste
 - seq64::seqevent, 310
- start_playing
 - seq64::mainwnd, 188
 - seq64::perfedit, 238
 - seq64::perform, 262
- stock_event_string
 - seq64::editable_event, 85

- stop
 - seq64::jack_assistant, [151](#)
 - seq64::mastermidibus, [195](#)
 - seq64::perform, [257](#)
- stop_key
 - seq64::perform, [263](#)
- stop_playing
 - seq64::mainwnd, [189](#)
 - seq64::perform, [263](#)
- stream_event
 - seq64::sequence, [352](#)
- stretch_selected
 - seq64::sequence, [353](#)
- string_is_void
 - seq64, [54](#)
- string_not_void
 - seq64, [53](#)
- string_to_midibyte
 - seq64, [53](#)
- string_to_pulses
 - seq64, [52](#)
- strings_match
 - seq64, [54](#)
- sync
 - seq64::jack_assistant, [155](#)
- sysex
 - seq64::mastermidibus, [197](#)
 - seq64::midibus, [214](#)
- tempo_from_beats_per_minute
 - seq64, [55](#)
- tempo_to_bytes
 - seq64, [55](#)
- text_x
 - seq64::user_settings, [385](#)
- text_y
 - seq64::user_settings, [385](#)
- ticks_to_delta_time_us
 - seq64, [56](#)
- time_as_measures
 - seq64::editable_event, [84](#)
- time_as_minutes
 - seq64::editable_event, [84](#)
- timeout
 - seq64::mainwid, [179](#)
 - seq64::perfedit, [238](#)
 - seq64::seqedit, [304](#)
- timer_callback
 - seq64::mainwnd, [188](#)
- timestamp
 - seq64::editable_event, [84](#)
- timestamp_format_t
 - seq64::editable_event, [82](#)
- timestamp_measures
 - seq64::editable_event, [82](#)
- timestamp_pulses
 - seq64::editable_event, [82](#)
- timestamp_time
 - seq64::editable_event, [82](#)
- timestring_to_pulses
 - seq64, [52](#)
- to_string
 - seq64, [60](#)
- toggle_playing
 - seq64::mainwnd, [189](#)
 - seq64::perfedit, [238](#)
- toggle_queued
 - seq64::sequence, [343](#)
- transpose_notes
 - seq64::sequence, [356](#)
- triggers
 - seq64::triggers, [364](#)
- type
 - seq64::keybindentry, [158](#)
- undo
 - seq64::perfedit, [238](#)
- unmodify
 - seq64::event_list, [106](#)
- unpaint_all
 - seq64::sequence, [354](#)
- unselect
 - seq64::sequence, [354](#)
 - seq64::triggers, [367](#)
- unselect_triggers
 - seq64::sequence, [347](#)
- unset_mode_group_learn
 - seq64::perform, [256](#)
- unset_sequence_control_status
 - seq64::perform, [260](#)
- update_background
 - seq64::seqroll, [327](#)
- update_markers
 - seq64::mainwid, [178](#)
- update_pixmap
 - seq64::seqtime, [332](#)
- update_sequences_on_window
 - seq64::mainwid, [178](#)
- update_sizes
 - seq64::perfroll, [276](#)
 - seq64::seqdata, [294](#)
 - seq64::seqevent, [309](#)
 - seq64::seqroll, [326](#)
- update_window_title
 - seq64::mainwnd, [189](#)
- v_adjustment
 - seq64::eventedit, [113](#), [114](#)
- valid_sequence
 - seq64::mainwid, [178](#)
- value_to_name
 - seq64::editable_event, [83](#)
- varinum_size
 - seq64::midifile, [225](#)
- verify_and_link
 - seq64::event_list, [107](#)
 - seq64::sequence, [354](#)
- versiontext

- seq64, [68](#)
- WHITE
 - seq64::font, [129](#)
- write
 - seq64::midifile, [219](#)
 - seq64::optionsfile, [233](#)
 - seq64::userfile, [393](#)
- write_byte
 - seq64::midifile, [223](#)
- write_long
 - seq64::midifile, [222](#)
- write_options_files
 - seq64, [60](#)
- write_prop_header
 - seq64::midifile, [224](#)
- write_proprietary_track
 - seq64::midifile, [225](#)
- write_seq_number
 - seq64::midifile, [224](#)
- write_short
 - seq64::midifile, [223](#)
- write_track_name
 - seq64::midifile, [223](#)
- write_varinum
 - seq64::midifile, [223](#)
- x_to_w
 - seq64::seqevent, [310](#)
- xy_to_rect
 - seq64::seqdata, [295](#)
 - seq64::seqroll, [328](#)
- YELLOW_ON_BLACK
 - seq64::font, [129](#)
- zero_markers
 - seq64::sequence, [354](#)
- zoom
 - seq64::user_settings, [384](#)
- zoom_check
 - seq64::perfedit, [237](#)