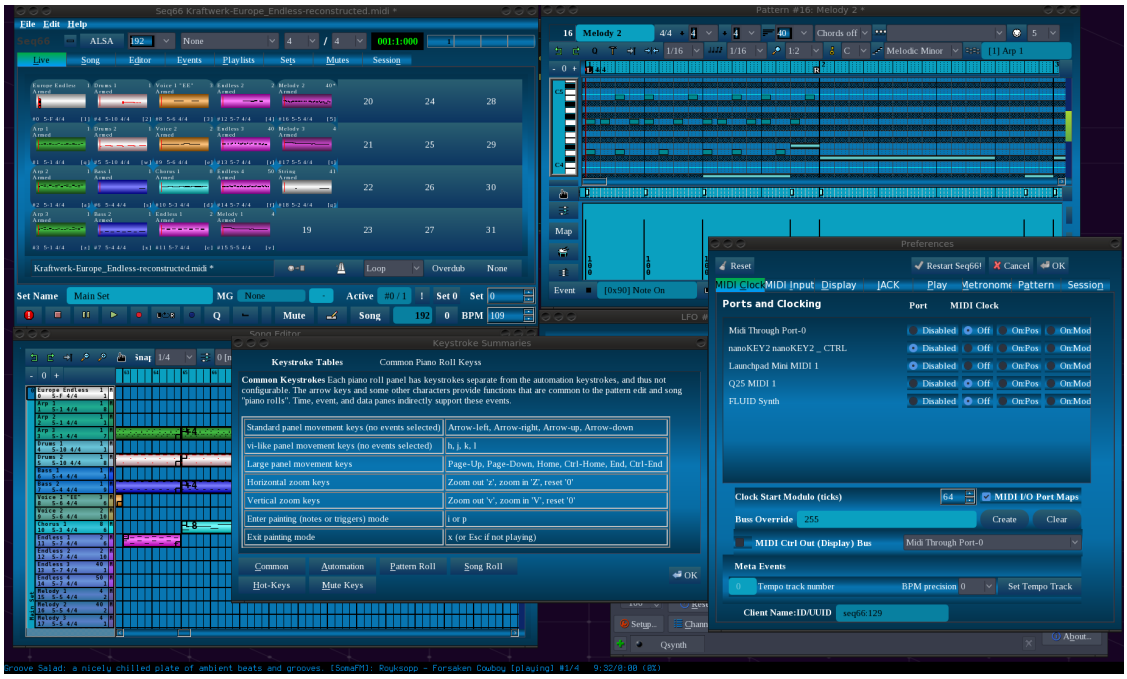
Seq66 User Manual v. 0.99.12

Chris Ahlstrom
(ahlstromcj@gmail.com)

January 13, 2024



Seq66 Windows with Perstfic Style-Sheet

Contents

1 Introduction **13**
1.1 Seq66: What!? . . . 13
1.2 Seq66: Why!? . . . 14
1.3 Improvements . . . 14
1.4 Document Structure . . . 15
1.5 Building Seq66 . . . 15

2 Let's Go! **15**
2.1 Device Changes . . . 16
2.2 Ports in Seq66 MIDI Files . . . 17
2.3 Windows . . . 17

3 Live Grid / Main Window and Tabs **17**
3.1 Tabs . . . 18
3.1.1 Tabs / Live . . . 19
3.1.2 Tabs / Song . . . 19
3.1.3 Tabs / Editor . . . 19
3.1.4 Tabs / Events . . . 19
3.1.5 Tabs / Playlist . . . 20
3.1.6 Tabs / Sets . . . 20
3.1.7 Tabs / Mutes . . . 20
3.1.8 Tabs / Session . . . 20
3.2 Main Top Controls . . . 20
3.2.1 Show/Hide Button . . . 21
3.2.2 PPQN Selection . . . 22
3.2.3 Buss Override for Play-set . . . 22
3.2.4 Global Beats Per Measure . . . 22
3.2.5 Global Beat Width . . . 23
3.2.6 BBT or HMS Time Display . . . 23
3.2.7 Beat Indicator . . . 23
3.3 Patterns Panel (Live Grid) . . . 23
3.4 Main Bottom Controls, First Row . . . 23
3.4.1 Set Name . . . 24
3.4.2 MG . . . 24
3.4.3 Underrun Indicator . . . 24
3.4.4 Set Reset ("!") . . . 24
3.4.5 Active Set Indicator . . . 24
3.4.6 Set 0 . . . 24
3.4.7 Set Changer . . . 25
3.5 Main Bottom Controls, Second Row . . . 25
3.5.1 Panic . . . 25
3.5.2 Stop . . . 25
3.5.3 Pause . . . 26
3.5.4 Play . . . 26
3.5.5 Record . . . 26
3.5.6 L/R Loop . . . 26

3.5.7 Live Song Record . . . 27
3.5.8 Keep Queue . . . 27
3.5.9 Mute Group Learn . . . 27
3.5.10 Developer Test . . . 28
3.5.11 Toggle Mute Status . . . 28
3.5.12 Song Editor . . . 28
3.5.13 Live/Song Mode . . . 28
3.5.14 PPQN Indicator . . . 28
3.5.15 BBT/HMS Toggle (removed) . . . 28
3.5.16 Tap BPM Button . . . 28
3.5.17 Beats Per Minute Control . . . 29

4 Menu **29**
4.1 Menu / File . . . 29
4.1.1 Menu / File / New . . . 31
4.1.2 Menu / File / Open . . . 31
4.1.3 Menu / File / Open Playlist . . . 31
4.1.4 Menu / File / Recent MIDI files . . . 32
4.1.5 Menu / File / Save and Save As . . . 33
4.1.6 Menu / File / Import / Import Project Configuration . . . 34
4.1.7 Menu / File / Import / Import MIDI to Current Set . . . 34
4.1.8 Menu / File / Import / Import Playlist . . . 34
4.1.9 Menu / File / Export / Export Project Configuration . . . 35
4.1.10 Menu / File / Export / Export Song as MIDI . . . 35
4.1.11 Menu / File / Export / Export MIDI Only . . . 35
4.1.12 Menu / File / Export / Export SMF 0 . . . 35
4.2 Menu / Edit . . . 35
4.2.1 Menu / Edit / Preferences . . . 37
4.2.1.1 Menu / Edit / Preferences / MIDI Clock . . . 37
4.2.1.2 Menu / Edit / Preferences / MIDI Input . . . 41
4.2.1.3 Menu / Edit / Preferences / Keyboard (removed) . . . 44
4.2.1.4 Menu / Edit / Preferences / Mouse (removed) . . . 44
4.2.1.5 Menu / Edit / Preferences / Display . . . 44
4.2.1.6 Menu / Edit / Preferences / JACK . . . 47
4.2.1.7 Menu / Edit / Preferences / Play Options . . . 48
4.2.1.8 Menu / Edit / Preferences / Metronome Options . . . 50
4.2.1.9 Menu / Edit / Preferences / Pattern . . . 51
4.2.1.10 Menu / Edit / Preferences / Session . . . 53
4.3 Menu / Help . . . 55
4.3.1 Menu / Help / About... . . . 55
4.3.2 Menu / Help / Build Info... . . . 55
4.3.3 Menu / Help / Song Summary File... . . . 55
4.3.4 Menu / Help / App Keys . . . 56
4.3.5 Menu / Help / Tutorial . . . 56
4.3.6 Menu / Help / User Manual . . . 56

5 Patterns Panel **57**
5.1 Patterns / Main Panel . . . 58
5.1.1 Pattern Slots . . . 58
5.1.1.1 Pattern Context Menus . . . 59
5.1.1.2 Pattern Labels . . . 64
5.1.2 Basic Pattern Control . . . 65
5.1.2.1 Pattern Queue . . . 65
5.1.2.2 Pattern Keep-Queue . . . 66
5.1.2.3 Pattern Oneshot . . . 66
5.1.2.4 Pattern Replace . . . 66
5.1.2.5 Pattern Solo . . . 67
5.1.2.6 Pattern Snapshot . . . 67
5.1.2.7 Pattern Learn . . . 67
5.1.3 Pattern Keys and Clicks . . . 68
5.1.3.1 Pattern Keys . . . 68
5.1.3.2 Other Pattern Clicks . . . 69
5.1.3.3 Metronome . . . 69
5.1.3.4 Count-in Recording . . . 70
5.1.3.5 Pattern Loop-Record Modes . . . 70
5.2 Patterns / Bottom Panel . . . 71
5.3 Patterns / Multiple Panels . . . 73
5.4 Patterns / Variable Set Size . . . 73
5.5 Patterns / Set Handling . . . 74

6 Pattern Editor **74**
6.1 Pattern Editor / First Row . . . 76
6.2 Pattern Editor / Second Row . . . 78
6.3 Pattern Editor / Measures Ruler . . . 83
6.4 Pattern Editor / Piano Roll . . . 85
6.4.1 Pattern Editor / Piano Roll Items . . . 86
6.4.2 Pattern Editor / Note Painting . . . 86
6.4.3 Pattern Editor / Note Editing . . . 87
6.4.4 Pattern Editor / Other Keys . . . 89
6.4.5 Pattern Editor / Zoom Keys . . . 89
6.5 Events Pane . . . 90
6.6 Left Buttons . . . 90
6.7 Data Pane . . . 91
6.8 Pattern Editor / Bottom Row . . . 94
6.9 Pattern Editor / Common Actions . . . 99
6.9.1 Pattern Editor / Common Actions / Scrolling . . . 100
6.9.2 Pattern Editor / Common Actions / Close . . . 100

7 Song Editor **100**
7.1 Song Editor / Playback Keystrokes . . . 102
7.2 Song Editor / Top Panel . . . 102
7.3 Song Editor / Measures Ruler . . . 104
7.4 Song Editor / Patterns (Names) Panel . . . 105
7.5 Song Editor / Song Roll . . . 106

7.5.1 Song Editor / Song Roll / Layout . . . 106
7.5.2 Song Editor / Song Roll / Keystrokes . . . 107
7.6 Song Editor / Bottom Panel . . . 108

8 Event Editor 108
8.1 Event Editor / Event Frame . . . 110
8.1.1 Event Frame / Data Items . . . 110
8.1.2 Event Frame / Navigation . . . 111
8.2 Event Editor / Info Panel . . . 111
8.3 Event Editor / Edit Fields . . . 112
8.4 Event Editor / Bottom Buttons . . . 115

9 Session Management 116
9.1 Session Management / Signals . . . 116
9.2 Session Management / JACK Session . . . 117
9.3 Seq66 Session Management / NSM . . . 118
9.3.1 Session Management / NSM / First Run Without NSM . . . 118
9.3.2 Seq66 Session Management / NSM / Run in NSM . . . 120
9.3.3 Session Management / NSM / Run with Remote NSM . . . 122
9.3.4 Session Management / Sessions Tab . . . 122
9.3.5 Seq66 Session Management / NSM / File Menu . . . 125
9.3.6 Seq66 Session Management / NSM / Debugging . . . 126
9.4 Seq66 Session Management / LASH . . . 127
9.5 Seq66 Session Management / sessions.rc . . . 127

10 Import/Export 128
10.1 File / Import Menu . . . 128
10.1.1 Import MIDI to Current Set . . . 128
10.1.2 Import Project Configuration . . . 129
10.1.3 Import Playlist . . . 129
10.2 File / Export Menu . . . 129
10.2.1 Export Project Configuration . . . 130
10.2.2 Export Song . . . 130
10.2.3 Export MIDI Only . . . 131
10.2.4 Export SMF 0 . . . 132

11 Seq66 Recording In Depth 132
11.1 Recording Scenarios . . . 132
11.1.1 Standard Recording . . . 132
11.1.2 Route-By-Buss . . . 133
11.1.3 Record-By-Channel . . . 133
11.2 Recording Modes . . . 134

12 Seq66 Configuration 134
12.1 Configuration File Commonalities . . . 135
12.1.1 [Seq66] Section . . . 136
12.1.2 [comments] Section . . . 136
12.1.3 Numeric Settings . . . 136
12.1.4 Boolean Settings . . . 136

12.1.5 Variables . . . 137
12.1.6 Stanzas . . . 137
12.2 Command Line . . . 137
12.3 'rc' File . . . 141
12.3.1 'rc' File / MIDI Control . . . 142
12.3.2 'rc' File / Mute Groups . . . 142
12.3.3 'rc' File / Color Palette . . . 142
12.3.4 'rc' File / Style Sheet . . . 143
12.3.5 'rc' File / Note Mapper . . . 145
12.3.6 'rc' File / MIDI-Clock Section . . . 146
12.3.7 'rc' File / MIDI Clock Mod Ticks . . . 147
12.3.8 'rc' File / MIDI File Tweaks . . . 147
12.3.9 'rc' File / MIDI-Meta-Events Section . . . 148
12.3.10 'rc' File / MIDI Input . . . 148
12.3.11 'rc' File / Manual (Virtual) Ports . . . 149
12.3.12 'rc' File / Port Map . . . 149
12.3.13 'rc' File / Keyboard Control Section . . . 149
12.3.14 'rc' File / JACK Transport . . . 150
12.3.15 'rc' File / Reveal Ports . . . 150
12.3.16 'rc' File / Metronome . . . 151
12.3.17 'rc' File / Count-In Recording . . . 151
12.3.18 'rc' File / Interaction Method . . . 152
12.3.19 'rc' File / Auto Option Save . . . 152
12.3.20 'rc' File / Last Used Directory . . . 152
12.3.21 'rc' File / Recent Files . . . 153
12.3.22 'rc' File / Play-List . . . 153
12.4 'usr' File . . . 153
12.4.1 'usr' File / MIDI Bus Definitions . . . 156
12.4.2 'usr' File / MIDI Instrument Definitions . . . 157
12.4.3 'usr' File / User Interface Settings . . . 158
12.4.4 'usr' File / User MIDI PPQN . . . 159
12.4.5 'usr' File / User Randomization . . . 159
12.4.6 'usr' File / User MIDI Settings . . . 160
12.4.7 'usr' File / User Options . . . 161
12.4.8 'usr' File / Additional Options . . . 161
12.4.8.1 'usr' File / Additional Options / [user-ui-tweaks] . . . 161
12.4.8.2 'usr' File / Additional Options / [user-session] . . . 162
12.4.8.3 'usr' File / Additional Options / [new-pattern-editor] . . . 163
12.5 'ctrl' File . . . 163
12.5.1 'ctrl' File / MIDI Control Settings . . . 164
12.5.2 'ctrl' File / Loop Control . . . 164
12.5.3 'ctrl' File / Mute-Group Control . . . 166
12.5.4 'ctrl' File / Automation Control . . . 166
12.5.4.1 Automation / BPM Up and Down . . . 168
12.5.4.2 Automation / Screen-Set Up and Down . . . 168
12.5.4.3 Automation / Mod Replace . . . 168
12.5.4.4 Automation / Mod Snaphot . . . 168
12.5.4.5 Automation / Mod Queue . . . 168

12.5.4.6 Automation / Mute Group ("Group Mute") . . . 169
12.5.4.7 Automation / Group Learn . . . 169
12.5.4.8 Automation / Playing Set . . . 169
12.5.4.9 Automation / Playback . . . 169
12.5.4.10 Automation / Song Record . . . 169
12.5.4.11 Automation / Solo . . . 169
12.5.4.12 Automation / Theu . . . 169
12.5.4.13 Automation / BPM Page Up and Page Down . . . 169
12.5.4.14 Automation / Set a Set . . . 170
12.5.4.15 Automation / Loop Mode . . . 170
12.5.4.16 Automation / Quan Record . . . 170
12.5.4.17 Automation / Reset Sets . . . 170
12.5.4.18 Automation / One-shot . . . 170
12.5.4.19 Automation / FF, Rewind, and Top . . . 170
12.5.4.20 Automation / Playlist Commands . . . 170
12.5.4.21 Automation / Tap BPM . . . 170
12.5.4.22 Automation / Start . . . 171
12.5.4.23 Automation / Stop . . . 171
12.5.4.24 Automation / Toggle Mute . . . 171
12.5.4.25 Automation / Song Position . . . 171
12.5.4.26 Automation / Keep Queuue . . . 171
12.5.4.27 Automation / Slot Shift . . . 171
12.5.4.28 Automation / Record Modes and Quantization . . . 171
12.5.4.29 Automation / BBT/HMS and LR Loop . . . 172
12.5.4.30 Automation / Undo and Redo . . . 173
12.5.5 Automation / More MIDI Control . . . 173
12.5.6 'ctrl' File / MIDI Control Output . . . 173
12.5.7 'ctrl' File / Macro Control Output . . . 174
12.5.8 'ctrl' File / Keyboard / Default Assignments . . . 175
12.5.9 'ctrl' File / AZERTY and QWERTZ Keyboards . . . 177
12.6 'mutes' File . . . 181
12.7 'drums' File . . . 184
12.8 'palette' File . . . 184
12.9 'playlist' File . . . 184

13 Seq66 Play-Lists 184
13.1 Seq66 Play-Lists / 'playlist' File Format . . . 185
13.2 Seq66 Play-Lists / 'rc' File . . . 186
13.3 Seq66 Play-Lists / 'ctrl' File / [midi-control] . . . 187
13.4 Seq66 Play-Lists / Command Line Invocation . . . 187
13.5 Seq66 Play-Lists / Verification . . . 188
13.6 Seq66 Play-Lists / User Interface . . . 188
13.6.1 Seq66 Play-Lists / User Interfaces / Playlist Buttons . . . 189
13.6.2 Seq66 Play-Lists / User Interfaces / Song Buttons . . . 190
13.6.3 Seq66 Play-Lists / User Interfaces / Info Fields . . . 191
13.6.4 Seq66 Play-Lists / Creating a Playlist File . . . 191
13.6.5 Seq66 Play-Lists / Playlist Sample . . . 192

14 Seq66 Set Master **192**
14.1 Set Handling . . . 194
14.1.1 Set Management . . . 194
14.1.2 Empty Set Handling . . . 194

15 Seq66 Mutes Master **195**
15.1 Mute Group . . . 195
15.2 Mute Master Tab . . . 196
15.3 Mute-Groups Table . . . 197
15.4 Mute-Groups Buttons . . . 197
15.5 Group-Patterns Buttons . . . 197
15.6 Other Controls . . . 198

16 Palettes for Coloring **199**
16.1 Palettes Setup . . . 199
16.1.1 Palettes Setup / Pattern . . . 200
16.1.2 Palettes Setup / Ui and Inverse Ui . . . 200
16.1.3 Palettes Setup / Brushes . . . 201
16.2 Palettes Summary . . . 201
16.3 Theming . . . 201

17 Seq66 Keyboard and Mouse Actions **202**
17.1 Keyboard Control . . . 202
17.2 Main Window . . . 205
17.3 Performance Editor Window . . . 206
17.3.1 Performance Editor Piano Roll . . . 206
17.3.2 Performance Editor Time Section . . . 207
17.3.3 Performance Editor Names Section . . . 208
17.4 Pattern Editor Piano Roll Keystrokes . . . 208
17.4.1 Pattern Editor Piano Roll . . . 208
17.4.2 Pattern Editor Event Panel . . . 210
17.4.3 Pattern Editor Data Panel . . . 210
17.4.4 Pattern Editor Virtual Keyboard . . . 210
17.5 Event Editor . . . 210

18 Seq66 In Windows **210**
18.1 Windows / Seq66 Installation . . . 211
18.2 Windows / MIDI Mapper . . . 211
18.3 Windows / Virtual Ports . . . 212
18.4 Windows / Seq66 Startup . . . 213
18.5 Windows / Keyboard Issues . . . 216

19 ALSA **216**
19.1 ALSA / Through Ports . . . 216
19.2 ALSA / Virtual MIDI Devices . . . 218
19.3 ALSA / Trouble-Shooting . . . 218
19.3.1 ALSA / Trouble-Shooting / ALSA Breakage . . . 218
19.3.2 ALSA / Trouble-Shooting / MIDI Clock . . . 219
19.3.2.1 ALSA MIDI Clock Send . . . 219

19.3.2.2 ALSA MIDI Clock Receive . . . 219
19.3.2.3 VMPK Issues . . . 219

20 JACK **220**
20.1 JACK / Transport . . . 220
20.2 JACK / Native MIDI . . . 222
20.2.1 JACK / MIDI Output . . . 222
20.2.2 JACK / MIDI Input . . . 223
20.2.3 JACK / MIDI Virtual Ports . . . 223
20.2.4 JACK / MIDI and a2jmidid . . . 223
20.3 JACK / Trouble-Shooting . . . 224
20.3.1 JACK / Trouble-Shooting / MIDI Clock . . . 224
20.3.1.1 JACK MIDI Clock Send . . . 224
20.3.1.2 JACK MIDI Clock Receive . . . 225
20.3.2 JACK / Trouble-Shooting / JACK Ports . . . 225
20.4 JACK / QJackCtl . . . 226
20.5 JACK / PulseAudio . . . 226

21 Port Mapping **227**
21.1 Input Port Mapping . . . 229
21.2 Output Port Mapping . . . 230
21.3 Port Mapping Example . . . 231
21.4 Port Setting SeqSpec . . . 232

22 Seq66 Headless Version **233**
22.1 Seq66 Headless Setup . . . 233
22.2 Seq66 Headless Stopping . . . 236

23 Launchpad Mini **236**
23.1 Launchpad Mini Basics . . . 237
23.2 System Survey, ALSA . . . 239
23.3 Control Setup . . . 239
23.3.1 Input Control Setup . . . 240
23.3.1.1 [loop-control] . . . 240
23.3.1.2 [mute-group-control] . . . 241
23.3.1.3 [automation-control] . . . 241
23.3.2 Output Control Setup . . . 242
23.3.2.1 [midi-control-out] . . . 242
23.3.2.2 [mute-control-out] . . . 242
23.3.2.3 [automation-control-out] . . . 243
23.4 Test Run, ALSA . . . 244
23.5 System Survey, JACK . . . 245

24 Concepts **246**
24.1 Concepts / Reload Session . . . 247
24.2 Concepts / Terms . . . 247
24.2.1 Concepts / Terms / loop, pattern, track, sequence . . . 247
24.2.2 Concepts / Terms / armed, muted . . . 247
24.2.3 Concepts / Terms / bank, screenset, play-screen . . . 248

24.2.4 Concepts / Terms / buss, bus, port . . . 248
24.2.5 Concepts / Terms / performance, song, trigger . . . 248
24.2.6 Concepts / Terms / Auto-step, Step-Edit . . . 248
24.2.7 Concepts / Terms / export . . . 249
24.2.8 Concepts / Terms / group, mute-group . . . 249
24.2.9 Concepts / Terms / play-set . . . 249
24.2.10 Concepts / Terms / PPQN, pulses, ticks, clocks, divisions . . . 249
24.2.11 Concepts / Terms / queue, keep queue, snapshot, oneshot . . . 249
24.3 Concepts / Sound Subsystems . . . 250
24.3.1 Concepts / Sound Subsystems / ALSA . . . 250
24.3.2 Concepts / Sound Subsystems / PortMIDI . . . 250
24.3.3 Concepts / Sound Subsystems / JACK . . . 250

25 MIDI Format and Other MIDI Notes 250
25.1 Standard MIDI Format . . . 251
25.1.1 Standard MIDI Format 0 . . . 251
25.2 Sequencer-Specific Meta-Events Format . . . 251
25.2.1 SeqSpec c_midibus . . . 252
25.2.2 SeqSpec c_midiinbus . . . 253
25.2.3 SeqSpec c_midichannel . . . 254
25.2.4 SeqSpec c_midiclocks . . . 254
25.2.5 SeqSpec c_triggers . . . 254
25.2.6 SeqSpec c_notes . . . 254
25.2.7 SeqSpec c_timesig . . . 255
25.2.8 SeqSpec c_bpmtag . . . 255
25.2.9 SeqSpec c_triggers_ex . . . 256
25.2.10 SeqSpec c_trig_transpose . . . 256
25.2.11 SeqSpec c_mutegroups . . . 256
25.2.12 SeqSpec c_gap_(ABCDEF) . . . 257
25.2.13 SeqSpec c_midictrl . . . 257
25.2.14 SeqSpec c_musickey . . . 258
25.2.15 SeqSpec c_musicscale . . . 258
25.2.16 SeqSpec c_backsequence . . . 258
25.2.17 SeqSpec c_transpose . . . 258
25.2.18 SeqSpec c_perf_bp_mes . . . 259
25.2.19 SeqSpec c_perf_bw . . . 259
25.2.20 SeqSpec c_tempo_map . . . 259
25.2.21 SeqSpec c_reserved_(12) . . . 259
25.2.22 SeqSpec c_tempo_track . . . 260
25.2.23 SeqSpec c_seq_color . . . 260
25.2.24 SeqSpec c_seq_edit_mode . . . 260
25.2.25 SeqSpec c_seq_loopcount . . . 260
25.3 MIDI Information . . . 261
25.3.1 MIDI Variable-Length Value . . . 261
25.3.2 MIDI Track Chunk . . . 261
25.3.3 MIDI System Events . . . 261
25.3.4 MIDI Meta Events . . . 262
25.3.5 Sequence Number (0x00) . . . 264

25.3.6 Track/Sequence Name (0x03) . . . 264
25.3.7 End of Track (0x2F) . . . 264
25.3.8 Set Tempo Event (0x51) . . . 265
25.3.9 Time Signature Event (0x58) . . . 265
25.3.10 SysEx Event (0xF0) . . . 266
25.3.10.1 Single SysEx Message . . . 267
25.3.10.2 Continuation Events . . . 267
25.3.10.3 Escape Sequence . . . 267
25.3.11 Universal SysEx Events (0xF0 0x7E and 0xF0 0x7F) . . . 268
25.3.11.1 Seq66 SysEx Handling . . . 268
25.3.12 Non-Specific End of Sequence . . . 269
25.4 More MIDI Information . . . 269
25.4.1 MIDI File Header, MThd . . . 269
25.4.2 MIDI Track, MTrk . . . 270
25.4.3 Channel Events . . . 270
25.4.4 Meta Events Revisited . . . 271

26 Kudos **271**

27 Summary **273**

28 References **273**

List of Figures

1 Seq66 Main Screen, Annotated . . . 18
2 Main Window Top Controls . . . 20
3 Main Window Tiny (shown full size) . . . 21
4 Main Window Bottom, First Row . . . 23
5 Main Window Bottom, Second Row . . . 25
6 Seq66 File Menu Plus Import/Export, Composite View . . . 30
7 File / Open . . . 31
8 File / Open Playlist . . . 32
9 Seq66 Menu File Recent Files . . . 32
10 Seq66 Menu File Recent Files, Full Paths . . . 32
11 File / Save As . . . 33
12 MIDI Clock (Output) Tab . . . 38
13 MIDI Input Tab . . . 42
14 MIDI Virtual Inputs . . . 43
15 Display Options . . . 45
16 Edit / Preferences / JACK . . . 47
17 Play Options . . . 49
18 Metronome Options . . . 50
19 Pattern Options . . . 52
20 Session Options . . . 54
21 Patterns Panel Drag-and-Drop . . . 57
22 Multiple Live Grids . . . 60
23 Patterns Panel Pop-up Menu . . . 61

24 Input Bus Popup Menu . . . 63
25 External Pattern Editor Window . . . 74
26 Pattern Editor Window, Annotated . . . 76
27 Pattern Fix Dialog . . . 80
28 Setting a Time Signature . . . 84
29 Virtual Keyboard Number and Note Views . . . 86
30 Data Pane Tempo . . . 92
31 Data Pane Tempo Line Draw . . . 92
32 Data Pane Tempo Events Drawn . . . 92
33 Data Pane Tempo Events Altered . . . 93
34 Data Pane Program Change . . . 93
35 Data Pane Note Grab-Handles . . . 94
36 Pattern Editor Event Button Context Menu . . . 95
37 Pattern Editor LFO . . . 96
38 One-Shot Pattern Recording . . . 98
39 Song Editor Window, Annotated . . . 101
40 Event Editor Window . . . 109
41 Event Category List . . . 112
42 Meta Event List . . . 113
43 System Message List . . . 114
44 MIDI File Unexportable . . . 130
45 MIDI File Layout Before/After Export . . . 131
46 Seq66 View with a Minimal Style-Sheet Applied . . . 143
47 Enhanced "Incrypt" Style-Sheet . . . 144
48 Enhanced "Incrypt" Style-Sheet Part 2 . . . 145
49 Seq66 Composite View of Native Devices . . . 154
50 Seq66 Composite View of Devices As Set in "sample.usr" . . . 158
51 Sets Tab . . . 193
52 New Sets Creation . . . 195
53 Mute Master Tab . . . 196
54 Seq66 First Startup in Windows . . . 213
55 'rc' File After Exiting First Startup . . . 214
56 MIDI Output Settings at Second Startup . . . 214
57 MIDI File Selection . . . 215
58 Opened MIDI File . . . 215
59 JACK MIDI Ports and Auto-Connect . . . 222
60 Clocks List Without Port Mapping . . . 228
61 Clocks List After Port Mapping . . . 229
62 Sample nanoKEY2 Control Setup . . . 234
63 Launchpad Mini Running with Seq66 . . . 245

List of Tables

1 Key Defaults. Control Keys . . . 176
2 Key Defaults. ASCII Keys 1 . . . 177
3 Key Defaults. ASCII Keys 2 . . . 178
4 Key Defaults. ASCII Keys 3 . . . 179

5 Key Defaults. Extended Keys 1 . . . 180
6 Key Defaults. Extended Keys 2 . . . 181
7 Key Defaults. Extended Keys 3 . . . 182
8 Key Defaults. Extended Keys 4 . . . 183
9 Main Window Support . . . 205
10 Performance Window Piano Roll . . . 206
11 Performance Editor Time Section . . . 207
12 Performance Editor Names Section . . . 208
13 Pattern Editor Piano Roll . . . 209
14 Pattern Editor Virtual Piano Keyboard . . . 210
15 Status Announcement Stanzas . . . 244
16 All SeqSpec Items . . . 253
17 MIDI Meta Event Types . . . 263
18 Application Support for Seq66 MIDI Files . . . 264

1 Introduction

Seq66 is a complete reboot of *Seq24* and a rewrite of *Sequencer64*. The following projects support *Seq66* and its documentation (see the **Help** menu):

- https://github.com/ahlstromcj/seq66.git
- https://ahlstromcj/github.io/docs/seq66/seq66-user-manual.pdf
- https://ahlstromcj.github.io/

Feel free to clone or fork it! I ain't gonna live forever! If you're in a hurry to get going, proceed directly to section 2 "Let's Go!" on page 15.

1.1 Seq66: What!?

Seq66 is a reboot of *Seq24*, a live-looping sequencer with an interface similar to a hardware sequencer, refactored for newer versions of *C++* for faster and simpler code. It drops the *Gtkmm* user-interface in favor of *Qt 5*, and has better handling of sets, mute-groups, sessions, configuration files, play-lists, and automation control. It supports the *Non/New Session Manager*, has a metronome function, multiple input port recording, a modifiable color and brush palette, and Qt style-sheets. Be prepared to note some significant differences between *Seq66* and our first reboot, *Sequencer64*. The basics are the same, though.

Seq66 is not a synthesizer. It requires a hardware or software synthesizer. It does not handle audio data, just MIDI. It works with *ALSA*, *JACK*, and *Windows*.

We have many contributors to acknowledge. Please see section 26 "Kudos" on page 271. If your name is not there, ping us!

1.2 Seq66: Why!?

The first reason to refactor *Sequencer64* is to take advantage of things learned in responding to user reports. The second reason is to use the new code as an opportunity to add new functionality such as *Non Session Manager* support. The third reason is to tighten the code by using newer features of *C++11* and later. The fourth reason is to make the innumerable minor improvements that come to attention with time and testing.

1.3 Improvements

The following improvements are some that have been made in *Seq66* versus *Sequencer64*.

- **Qt** 5 as the standard user-interface; a better live frame using Qt push-buttons; the main window's size can change; palette files; Qt style-sheets for control of colors and fonts for many user-interface items. A Qt linear-gradient default style for painting the progress boxes, notes, and triggers.
- Improved the **song editor** for laying out patterns; includes transposable triggers; can be opened in a tab or window.
- The **mutes editor** tab, improves mutes handling and control.
- A **playlist editor** tab with improved flexibility.
- A **sets editor** tab.
- A **events editor** tab. To trouble-shoot and fix minor event issues and add meta text events.
- A **session** tab (and Edit / Preference pages) to see that *Seq66* is running in the desired environment and configuation.
- **Non Session Manager** support. The sessions tab shows the locations of configuration files for the session.
- **MIDI control/display automation**. Control by "launchpad" devices, and display of statuses.
- Repartitioning of **configuration files** into separate files for flexibility; added a **color palette** file, Qt **style-sheets** (`*.qss`); an enhanced keystroke and **MIDI 'ctrl'** file, with support for displaying pattern and action statuses.
- Improved **alternate** keyboard layout support.
- **Mapping** of port names to a consistent set of port numbers.
- Providing for **routing of MIDI input** events to patterns based on port number or output-channel number.
- **Export songs** in SMF 0 and 1 formats.
- **Internals**: More efficient lookups for using control maps and lambda functions.
- Configurable as a **command-line** application with the option to run as a headless *daemon*.

For developers, a *Seq66* build is customizable via C macros and by enabling/disabling options at 'configure' time. Distro maintainers may create their own build configurations. We cannot show all permutations of settings in this document, so don't be surprised if some screenshots don't quite match one's setup.

1.4 Document Structure

The structure of this document follows the user-interface of *Seq66*. To help the reader jump around this document, it provides multiple links, references, and index entries.

1.5 Building Seq66

There are a number of ways of building Seq66.

- **Autotools Build and Install**. Configure, make, and install.
- **Bootstrap Install**. Generate autotools files and build.
- **OpenSUSE and Fedora**. Specifics for that Linux distro.
- **Qmake-based Install**. Optional on Linux, mandatory for Windows.
- **Arch Linux**. There is a nice *AUR* package and some other packages noted at ([29]).

The `INSTALL` file included with the source code and documentation goes into great detail about these methods. Also see `data/readme.windows` and `contrib/notes/git.txt` for supplemental information.

2 Let's Go!

Seq66 requires ALSA or JACK on Linux; it does not support Pipewire. It might work with a Pipewire-JACK setup, but that usage has not been tested. Another aspect to look into is interactions with PulseAudio-pw. On Windows, *Seq66* uses the *MultiMedia API* with a wrapper that is our modified version of *PortMIDI* [27]. Theoretically that library should support *Mac OSX*, but we have no Mac to build and test with. This manual assumes that one is running *Seq66* on Linux, for the most part. But see section 18 "Seq66 In Windows" on page 210.

For the first run, make sure all desired MIDI devices are plugged in and that all desired software synthesizers are running. Only ports present at startup will appear; this version of *Seq66* does not support automatic handling of port changes at run-time. In addition, port-mapping (see section 21 "Port Mapping" on page 227) is enabled by default; this makes the port setup more flexible..

Start *Seq66*, verify that it comes up, and then exit it immediately. This first start creates the configuration files in

```
/home/user/.config/seq66/}                      (Linux)
C:/Users/your\_user\_name/AppData/Local/seq66}   (Windows)
```

If a session manager (e.g. *NSM*) is used, the configuration directory is determined by the session manager, and is also called a "session" directory.. The main configuration file is `qseq66.rc` (Linux) or `qpseq66.rc` (Windows). The '.rc' file contains ports and port-mapping for the devices initially on the system. It also specifies the names of a number of other configuration files.

Now start *Seq66* to use JACK or ALSA for MIDI. For USB MIDI devices on JACK, the `a2jmidid` program might need to be running (see section 20 "JACK" on page 220, for details.) On *Windows*, just run `qpseq66.exe`; see section 18 "Seq66 In Windows" on page 210.

For better trouble-shooting, run it via command-line in a terminal window, at first and provide a MIDI file, to see what messages might appear. Also note that a log file is created; by default message will be written to it. The port settings will depend on your system. On our system, the synthesizer (*Yoshimi*) comes up on MIDI buss 5. (On *Windows*, buss 0 is the "MIDI Mapper", while buss 1 is the built-in wavetable synthesizer, which is normally under control of buss 0.) The `--buss` option remaps all events to the desired buss for the current run:

```
$ qseq66 --alsa --buss 5 /usr/share/seq66/midi/b4uacuse-gm-patchless.midi
$ qseq66 --jack --buss 5 /usr/share/seq66/midi/b4uacuse-gm-patchless.midi
```

The buss-override setting can also be made via the port drop-down control in the main window. It will modify the MIDI file. The "data" directory is a directory created upon installation of the application:

```
/usr/share/seq66-0.99/                    (Linux)
C:/Program Files/Seq66/data/              (Windows)
```

It contains sample MIDI files and configuration files. Some of the files in those directories apply to both operating systems, so be sure to look at them. The user configration files are stored in the user's "home" area:

```
/home/username/.config/seq66/qseq66.*     (Linux)
C:/Users/username/AppData/Local/seq66     (Windows)
~/NSM Sessions/sessname/seq66.nXXXX/config (NSM)
```

The configuration files in the "home" directory are created after the first run of *Seq66*. If all is well, the "play" button will start playback and the tune sounds. If not, look into **Edit** / **Preferences** or the configuration files. Don't change the configuration files while *Seq66* is running, as they are generally re-saved at application exit.

Bug? The first-start change of input port settings might not get saved to the 'rc' file. The workaround is to exit and rerun *Seq66*, and make the setting again. Mysterious.

2.1 Device Changes

After running *Seq66* and getting it to work, one might either add new MIDI devices and software synthesizers, or remove them. If port-mapping is enabled (the default, and recommended), then one can go to **Edit** / **Preferences** / **MIDI Clock** and click the **Make Maps** button. This action stores (again) the current set of MIDI devices. Restart *Seq66* and the new set of devices should appear.

Seq66 will also detect at startup if there are unavailable ports or if there are now more real MIDI

ports than mapped ports. It will prompt for a potential remapping and restart. Click that button if feeling lucky, otherwise exit and fix the 'rc' file with a text editor.

Warning: If one has hand-tailored the port maps to represent all possible devices one has, the remap or restart buttons might **remove** some of those devices. Always keep a **backup** of a known-good 'rc' file in a safe place.

In version 2 of *Seq66*, a year or two from now, we hope to make the adjustments automatic.

2.2 Ports in Seq66 MIDI Files

When opening a pre-existing *Seq66* which contains patterns with output port numbers that don't exist on the current system, a prompt will appear with option to remap-and-restart *Seq66*. However, what normally needs to be done is to change the port numbers to the numbers of existing ports, and then save the MIDI file. One way to do this is to use the "buss-override" feature (see section 3.2.3 "Buss Override for Play-set" on page 22), which affects all patterns shown on the Live grid.

2.3 Windows

Details about *Windows* are in the file `C:/Program Files/Seq66/data/midi/readme.windows` and in section 18 "Seq66 In Windows" on page 210. The first-start depends on if it is a bare *Windows* install, or if supplemental MIDI support such as the *CoolSoft MIDIMapper* and *VirtualMIDISynth* have been installed. If they haven't, the default MIDI setup will contain no MIDI inputs, and two MIDI outputs, the *MIDI Mapper* and the *Microsoft GS Wavetable* synthesizer. The former grabs control of the latter, making it unavailable! *Seq66* detects this situation and enables only the MIDI Mapper.

Therefore, run *Seq66* for the first time, then exit it and restart it. One should then be able to play MIDI files to the MIDI Mapper, if nothing else. We hope to get some input from *Windows* users, as our main usage is *Linux* and we run *Windows* only in an old virtual machine.

As for virtual ports in *Windows*, a useful too is *LoopMIDI* ([12]), as noted in the section on *Windows*.

3 Live Grid / Main Window and Tabs

For our walk-through of the main window, the following figure shows it using blue labels for reference.

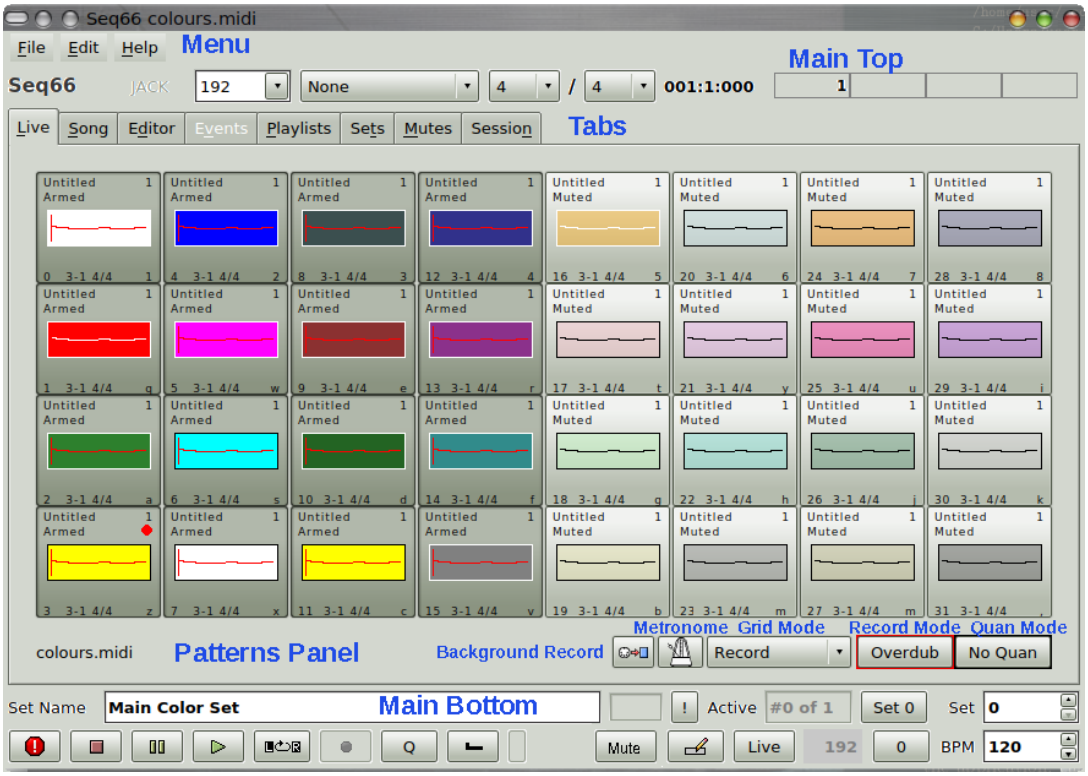


Figure 1: Seq66 Main Screen, Annotated

The *Seq66* main window (i.e. the "Live Grid") appears as shown above. This figure has many differences from the *Seq24* main window, but the functionality is similar. *Seq66* allows resizing, and can be configured to start with its size scaled up or down. There is also a button near the top (not shown) that can hide the menu and the bottom panels, to allow reducing the size of the window event further. Its features, including the "look" of the application, can be configured via the 'rc', 'usr', 'ctrl', 'drums', 'playlist', 'mutes', and 'palette' configuration files, via command-line options, via desktop themes, and via Qt ('qss') style-sheets. We break the discussion into sections for the following groups shown in the figure above:

- **Tabs**
- **Main Top**
- **Main Bottom**
- **Menu**

3.1 Tabs

The following tabs are present in *Seq66*:

- **Live**
- **Song**
- **Editor**

- **Events**
- **Playlists**
- **Sets**
- **Mutes**
- **Session**

We present brief descriptions here and links to sections with more detail. All tabs are discussed in more detail, each in its own section.

3.1.1 Tabs / Live

The **Live** tab is foremost in the application. It provides a grid of *patterns* (also called *slots*, *loops*, *tracks*, or *sequences*) that display recorded MIDI data, status information, and provide popup-menus for each pattern. The buttons can be colored via a palette, and the status is easy to see from the coloring of activated buttons and a label that says "Armed" versus "Muted". The buttons can be toggled by a keystroke, shown in the lower right corner of the button. Another name for the **Live** tab is the **Patterns Panel**; it can be replicated in multiple external windows. See section 5 "Patterns Panel" on page 57.

3.1.2 Tabs / Song

The **Song Editor** combines all patterns into a complete tune with controlled repetitions of each pattern. It provides a way to lay out *triggers* for each pattern that control playback without musician interaction. The editing capabilities of the song editor are extensive, including dragging triggers, addition transposition to triggers, and more. See section 7 "Song Editor" on page 100.

3.1.3 Tabs / Editor

The **Pattern Editor** allows for the recording and editing of notes and other MDII events. It can be shown in the **Editor** tab or in an external window. When opened in the tab, the pattern editor is compressed vertically by reducing the height of the "data" area and removing some buttons. Otherwise, the pattern editor works the same in the tab and in external windows. See section 6 "Pattern Editor" on page 74.

3.1.4 Tabs / Events

The **Event Editor** provides a way to see the details of events and to do some modifying of them. It's meant for light usage; there are some less-used events that it cannot edit. It is especially useful in trouble-shooting a song. See section 8 "Event Editor" on page 108.

3.1.5 Tabs / Playlist

The **Playlist Editor** is meant for the creation of play-lists. One can create multiple playlists with multiple songs in each playlist. The selection of playlists and songs can be done via keystrokes or MIDI control. Note that editing of the playlists can also be done by directly editing a `.playlist` file. See section 13 "Seq66 Play-Lists" on page 184.

3.1.6 Tabs / Sets

The **Sets Editor** allows one to see, edit, and rearrange the sets that a tune contains. Sets are also referred to as "banks". A useful feature of sets is that they can selected and cause a major change in the patterns that are playing. See section 14 "Seq66 Set Master" on page 192.

3.1.7 Tabs / Mutes

Mute-groups provide a way to turn on and turn off multiple patterns at once via a keystroke or a MIDI control; the **Mutes Editor** also provides a way to use buttons to control the mute-group. Mute-groups can be saved in the `.mutes` file or in the tune itself. See section 15 "Seq66 Mutes Master" on page 195.

3.1.8 Tabs / Session

The **Session** tab displays some aspects of the running setup of *Seq66*. It displays the name of the session manager, if any, client IDs, etc. It also has a button reload the whole session after configuration changes are made.

Here, a session log file can also be specified. If defined, then all messages written to the terminal when *Seq66* is run from a terminal shell are instead redirected to this file. The default log file is `seq66.log` specified to use the "home" directory. This file can be cleared using the button to the right of it; however, messages will continue to be logged to this file until *Seq66* is completely exited and then manually restarted.

In addition, text data for patterns can be viewed and edited here. Select the desired pattern, enter the desired text, and click the **Save Info** button. The song is now modified and can be saved.

Also see section 9 "Session Management" on page 116.

3.2 Main Top Controls

Here, we first discuss the top and bottom **Main** controls, as shown in the following collapsed figure:

Figure 2: Main Window Top Controls

See the following section and section 3.4 "Main Bottom Controls, First Row" on page 23. The top panel of the Pattern window is simple, consisting of the name of the program and a few controls. The top main control items are, from left to right:

- **Seq66 Logo**
- **Show/Hide Button**
- **Engine Status**
- **PPQN Selection**
- **Buss Override for Play-set**
- **Global Beats Per Measure**
- **Global Beat Width**
- **BBT/HMS Time Display**
- **Beat Indicator Bar**

The status indicators that can appear to the right of the logo:

- **ALSA** appears if running using ALSA for MIDI.
- **JACK** appears if running using JACK for MIDI.
- **JACK Master** appears if running using JACK transport, as transport master. JACK transport can be used even if using ALSA for MIDI.
- **JACK Slave** appears if using JACK transport as transport slave.
- **Portmidi** appears if using the internal portmidi-derived MIDI engine. This is currently always the case on *Windows*.

3.2.1 Show/Hide Button

If present (it is a build option), this button at the left of the top bar allows the main menu and the bottom two rows of controls to be hidden. This measure can save a lot of space (e.g. in a *Raspberry Pi* setup. If your window manager (e.g. *Fluxbox*) allows hiding the window decorations, the smallest useful size of the main window can be about 450 x 320 pixels. Of course, a good 'ctrl' automation file (e.g. for the LaunchPad) is necessary to control *Seq66* in this tiny setup; main menu hot-keys are not accessible with the main menu hidden.

Figure 3: Main Window Tiny (shown full size)

Of course, most of the other tabs are not usable at this small size, the song editor menu, shortcut, and button aren't available, and the pattern editor still comes up big. Might be enough for live play? Perhaps we should add a 'usr' option for the "tiny" mode? And hide the rest of the tabs? Switch some buttons around?

3.2.2 PPQN Selection

This drop-down allows one to change the pulses-per-quarter-note (PPQN) of the loaded tune, and this change can then be saved, if desired, with the file. As with *Seq24*, the default PPQN is 192. This can be changed to to other values: `32, 48, 96, 192, 240, 384, 768, 960, 1920, 2400, 3840, 7680, 9600, and 19200`. Values, even weird ones, can be entered by typing them. If a MIDI file is loaded, this modifies that file, rescaling all the pattern events and pattern triggers. Also see section 4.2.1.7 "Menu / Edit / Preferences / Play Options" on page 48.

3.2.3 Buss Override for Play-set

This drop-down allows for overriding the buss (port) number used by all of the patterns in the current play-set. Unlike the `buss-override` setting in the 'usr' file (see section 12.4.6 "'usr' File / User MIDI Settings" on page 160), this action causes a modification of the file (and a prompt to save at exit).

The *play-set* is the current set of patterns to be played. Normally, this set holds only the active patterns in the current play-screen. However, it can also be configured to include patterns from other sets. (See section 12.3 "'rc' File" on page 141.)

The list of output busses is either the existing MIDI ports on the system, or, if port-mapping (see section 21 "Port Mapping" on page 227) is active, the list of mapped output ports. Port-mapping is a useful way to redirect the set to a different output device; it can be used to provide a full set of virtual devices that any of the user's sequences can depend on.

Another way to specify busses is the `--buss n` command-line option. It causes *every* pattern in *every* set in the MIDI file to be directed to that buss number, and when a new sequence/pattern is created. This option is only for convenience in testing. Save the file, and it will have that buss number as part of each track's data, which makes the song file less portable, so be careful with both options.

3.2.4 Global Beats Per Measure

This drop-down changes the global beats/measure for the song. Along with the beat-width setting, this set of values allows for a large number of different time signatures, even crazy ones. *Warning*: It modifies every pattern in the song, but is not otherwise saved in the configuration. Values: `1 to 16, 32`

3.2.5 Global Beat Width

This drop-down changes the global beat width (time-signature denominator) for the song. Along with the beats-per-measure setting, this set of values allows for a large number of different time signatures. *Warning*: It modifies every pattern in the song, but is not otherwise saved in the configuration. Values: `1 to 16, 32`

3.2.6 BBT or HMS Time Display

This push-button shows the current time during playback. It can be shown in BBT (bars:beats:ticks) or HMS (hours:minutes:seconds), which can be switched by clicking this button. (The **BBT/HMS** at the bottom of the window has been removed.)

3.2.7 Beat Indicator

The beat indicator is inspired by the *Kepler34* implementation. It shows the first beat in color, and the rest of the beats in the theme color. It does not adapt to changes in the time-signature until playback is stopped.

3.3 Patterns Panel (Live Grid)

In the center of the main window is the *patterns panel*, also known as the *live grid*, where patterns/loops/tracks are shown and where they can be controlled. A MIDI file can be dragged-and-dropped onto the live grid in order to open it. Also shown in the patterns panel are a background-record button, metronome button, a grid-mode dropdown to control how the grid is used, a button to choose how recording works (e.g. overdub versus merge), and a button to indicate how recording is quantized. The patterns panel is discussed in more detail in section 5 "Patterns Panel" on page 57.

3.4 Main Bottom Controls, First Row

The bottom main control items take up two rows. Here is the first row and its contents:

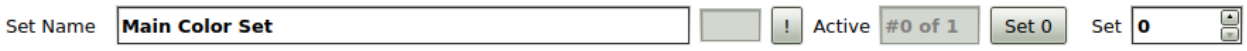


Figure 4: Main Window Bottom, First Row

- **Set Name**
- **MG**
- **Underrun Indicator**
- **Set Reset ("!")**
- **Active Set Indicator**
- **Set 0 Selector**
- **Set Changer**

3.4.1 Set Name

This text field shows the name of the current set, and also allows editing the set name.

3.4.2 MG

This text field shows the name of the current mute-group, if one is in force. Currently not shown in the figure above.

3.4.3 Underrun Indicator

This text field is empty under normal circumstances, including during playback. However, if one opens a pattern editor for a pattern with a lot of events, and turns on recording for that pattern, then this field might start flashing numbers, which indicate the current underrun value in microseconds. This gives an indication that timing of playback might be off a bit, usually less than 10 milliseconds. This happens because the event-list is locked during redrawing, when recording, to avoid problems when new notes come in.

3.4.4 Set Reset ("!")

This small button with an exclamation point to the right of the set name is meant to clear out all playing sets, and make only the current set the playing set. This button is useful when in "all-sets" mode or when sets get added automatically via the "additive" mode, so that multiple sets are playing at once. See section 12.3 "'rc' File" on page 141.

3.4.5 Active Set Indicator

This read-only text field shows the set number of the currently active set. One can open a number of external *Live Frames* by *shift-left-clicking* on pattern slots. The currently active set is then the set that has the mouse focus. This allows for working with multiple sets without a lot of mouse/keyboard navigation.

3.4.6 Set 0

This button provides a fast way to get back to set 0, without clicking the mouse a bunch of times. It is useful when creating an external live grid (see section 22 "Multiple Live Grids" on page 60), which currently also sets the main grid to the same set. Click on this button, and the external set and set 0 can be seen at the same time.

3.4.7 Set Changer

This spin-box allows showing a different set in the main windows. This set can be modified by adding new patterns, changing its name, or importing other MIDI files into the current set.

3.5 Main Bottom Controls, Second Row

Here is the second row and contents:

Figure 5: Main Window Bottom, Second Row

- **Panic**
- **Stop**
- **Pause**
- **Play**
- **Record** (not yet shown in the figure)
- **L/R Loop**
- **Live Song Record**
- **Keep Queue**
- **Mute Group Learn**
- **Developer Test**
- **Mute** (Toggle Mute Status)
- **Song Editor**
- **Live/Song**
- **PPQN Indicator**
- **Tap BPM**
- **Beats Per Minute Control**

Many of these controls have keystrokes and MIDI-control slots that can be set up in the 'ctrl' file.

Note that **BBT/HMS Toggle Button** has been removed (version 0.99.9). One can now click on the **BBT/HMS Time Display** to change the format.

3.5.1 Panic

This button causes playback to stop, all patterns to mute, and flushes the MIDI buss. There is a keystroke control and a MIDI control for this automation operation, plus a MIDI-announcement (output) configuration item for it.

3.5.2 Stop

This button stops playback and rewinds to the beginning of the song. By default, the `Esc` key operates this function, and there is both a MIDI-control slot and a MIDI-announcement slot available

for it.

3.5.3 Pause

This button stops playback, but does not rewind to the beginning of the song. It also resumes playback at the same point as the pause. By default, the `Period` key operates this function, and there is a MIDI-control slot and a MIDI-announcement slot available for it. This key is also hardwired to pause and start playback in the pattern editor and the song editor.

3.5.4 Play

This button starts playback, either at the beginning or at the pause point. Also called the "start button". By default, the `Space` key operates this function, and there is both a MIDI-control slot and a MIDI-announcement slot available for it. This key is also hardwired to toggle playback in the pattern editor and the song editor.

3.5.5 Record

This new button turns on recording for the various kinds of recording described in section 11 "Seq66 Recording In Depth" on page 132. Its color indicates the record-by feature in force:

- **Red**. This color indicates the normal recording mode, which toggles only a single pattern for recording.
- **Yellow**. This color indicates the record-by-channel recording mode, which toggles, for recording, all patterns that have an output channel set.
- **Green**. This color indicates the record-by-buss recording mode, which toggles, for recording, all patterns that have an input bus set.

These modes are selected in the 'rc' file as described in section 12.3.7 "'rc' File / MIDI Clock Mod Ticks" on page 147. They are mutually exclusive.

3.5.6 L/R Loop

This button has been added to the main window as of version 0.93.0 of *Seq66*. This reflects that the **L**/**R** loop markers in the song editor can now be used in the pattern editor as well. This new feature makes it easier to focus in on a pattern and tinker repeatedly with the same small section. In addition, looping can now be done in both the Live and Song modes of playback.

Activates loop mode. When play is activated, plays the song and loop between the **L marker** and the **R marker**. It is a state button, and its appearance indicates when it is depressed, and thus active. If this button is deactivated during playback, then playback continues past the **R marker**. Note that these markers can be placed using Left and Right mouse clicks, respectively, in the time/measures ruler. Also note that this button has been added to the main window, and

appears only in the "external" version of the song editor. Furthermore, the **L**/**R** markers can also be set in a pattern editor, where they can be used to focus in on a small section of notes. Lastly, this looping mechanism is also available in Live mode as well.

3.5.7 Live Song Record

This button causes a live playing session to be recorded to Song mode. That is, triggers are added to the song automatically as the musician mutes and unmutes patterns, and the triggers can then be seen as layouts in the *Song* editor. By default, the `P` key operates this function.

Using the `Ctrl` modifier while clicking this button turns off the recording-snap feature. This change can also be done in the song editor. Using the `Shift` modifier while clicking this button causes recording immediately after starting playback. This feature is useful in avoid delay at start up.

3.5.8 Keep Queue

Puts the application into a "sticky" queue mode. In this mode, pressing a pattern key does not do a mute/unmute function, but instead turns on queuing for the selected pattern. By default, the `Backslash` key operates this function, and there is a MIDI-control slot available for it.

3.5.9 Mute Group Learn

Also called the "L" button. Sets up to learn the current set of active patterns ("mute group") into a mute-group. The default keystroke to control this button is lower-case "L". When in group-learn mode, the `Shift` key cannot be hit, so the group-learn mode automatically converts the keys to their shifted versions. This feature known as *shift-lock* or *auto-shift*. This feature is not used if a non-QWERTY keyboard layout is specified in the 'ctrl' file.

After pressing the "L" button, the user can then press a keystroke, which is automatically shifted, and the pattern set is saved, and can be recalled by pressing that button, shifted, later. It can be saved in a 'mutes' file, as part of the MIDI tune, or in both places.

Example: We have 5 patterns armed in the current set. Press the "L" button, and then press the "s" key. These pattern statuses are saved and can be recalled later by the "S" ("s"-shifted) key.

By default, the `el` (lower-case "L") key also sets this function, and there is a MIDI-control slot available for it, as well as a MIDI-announcement slot. In addition to that, one can also press the `Ctrl-L` key. The "el" with it!

Remember that groups work with the playing ("in-view") screen-set. One must change the screenset and give it the command to make it the playing one. By default, the `Home` key is configured for this purpose.

There is also a setting in the 'mutes' file called `mute-group-selected`. If this value is set to a value from 0 to 31, then that mute group will be automatically applied when *Seq66* starts up. This is useful with the loading of the most-recent MIDI file (which is also a feature of *Seq66*. Also see the section about the "mute master" tab, section 15 "Seq66 Mutes Master" on page 195.

3.5.10 Developer Test

This button is always disabled. Functionality is added temporarily when testing new features. Ignore this button.

3.5.11 Toggle Mute Status

The **Mute** button toggles the mute/armed status of all of the patterns in the current set. Works the same as the **Edit / Toggle All Tracks** menu or the hard-wired `Ctrl-T` key.

3.5.12 Song Editor

This button (with a "pencil" icon) brings up an external window for editing the Song/Performance information. If already up, it closes it. Works the same as the **Edit / Song Editor** menu or the hard-wired `Ctrl-E` key.

3.5.13 Live/Song Mode

This button toggles between the *Live* and *Song* performance mode. In the Live mode, the musician controls are muting/unmuting of each pattern. In the Song mode, the triggers layed out in the **Song Editor** control the playback. By default, the `F10` key operates this function, There is also a MIDI automation control for this button.

3.5.14 PPQN Indicator

This read-only field displays the current PPQN for the current tune.

qseq66.usr: `[user-midi-settings] midi_ppqn`

3.5.15 BBT/HMS Toggle (removed)

The time display has been converted to a button, and click it will switch the format. So the BBT/HMS Toggle button *has been removed*.

3.5.16 Tap BPM Button

Tap this button with a regular beat to determine the beats-per-minute of the tapping. With each tap, the counter on the button increments and the BPM is recalculated. Stop tapping for a few seconds to reset the counter. By default, the `F9` key operates this function, but it STILL NEEDS WORK to show the results in the BPM control. There is also a MIDI-control slot for this function.

3.5.17 Beats Per Minute Control

This control can be text-edited or spun to change the beats/minute value used in playing back the current song. This value is also saved to the file.

4 Menu

The *Seq66* menu structure is more complex than that of *Seq24*. In particular, the *File* menu has two variants: a normal file menu, and a file menu when *Seq66* is running under the *New/Non Session Manager*. (See section 9.3 "Seq66 Session Management / NSM" on page 118.)

4.1 Menu / File

The **File** menu is used to save and load files in Standard MIDI Format 0 or 1, *Cakewalk* "WRK", and *Seq66* MIDI files. It also supports a list of recent files, and (new with version 0.98.3) sub-menus for import and export functions, which have expanded quite a bit.

The *Seq66* **File** menu contains the sub-items shown below. The next few sub-sections discuss the sub-items in the **File** menu. Please note that these entries are different if *Seq66* is started under the control of the *New/Non Session Manager*. See section 9.3.5 "Seq66 Session Management / NSM / File Menu" on page 125. However, the import and export menus remain the same, although there are slight differences in how they work.

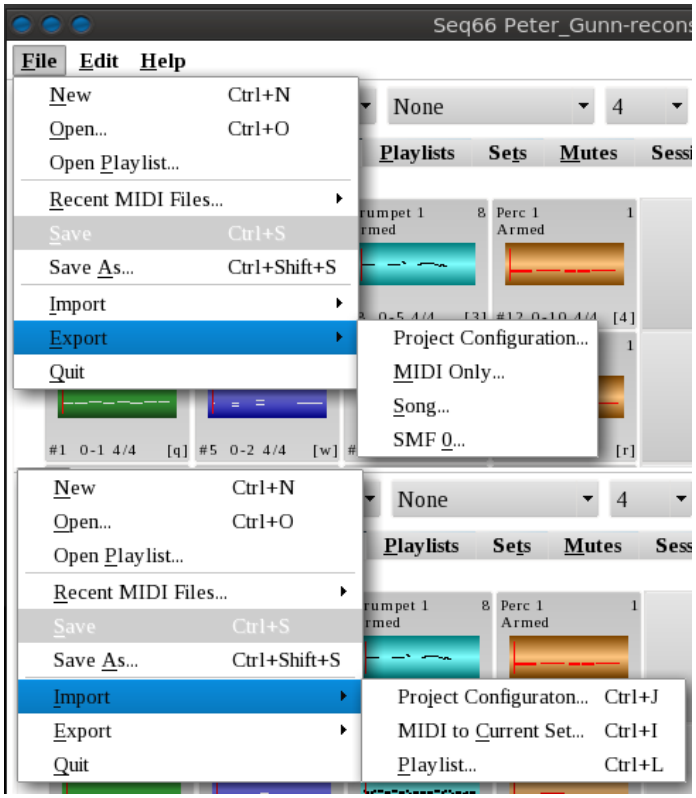


Figure 6: Seq66 File Menu Plus Import/Export, Composite View

1. **New**
2. **Open**
3. **Open Playlist**
4. **Recent MIDI files**
5. **Save**
6. **Save As**
7. **Import**
    1. **Project Configuration...**
    2. **MIDI to Current Set...**
    3. **Playlist...** Once the playlist is imported, *Seq66* is automatically ***restarted*** in order to load the playlist. Be careful!
8. **Export**
    1. **Project Configuration...**
    2. **MIDI Only...**
    3. **Song...**
    4. **SMF 0...**
9. **Quit** (**Exit** in *Windows*)

For information on the **File** menu when *Seq66* is running under the *Non Session Manager*, see section 9.3.5 "Seq66 Session Management / NSM / File Menu" on page 125.

4.1.1 Menu / File / New

The **New** menu entry clears the current song. (A play-list or mute-groups setup, if loaded, are not affected.) If unsaved changes are pending, the user is prompted to save the changes. Prompting for changes is more comprehensive than *Seq24*. However, when in doubt, save! Keep backups of your tunes and configuration files!

4.1.2 Menu / File / Open

The **Open** menu entry opens a song (MIDI file or *Cakewalk* WRK file), replacing the current song (after a prompt if the song was modified). It opens up a standard file dialog:

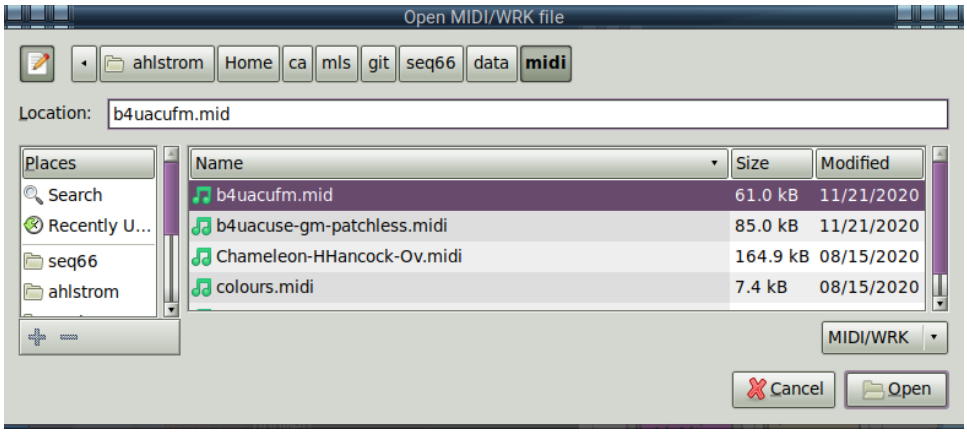


Figure 7: File / Open

This dialog lets one type a file-name, highlighting the first file that matches the characters typed. *Seq66* can open *Seq66*, MIDI SMF 0, and SMF 1 files, and *Cakewalk* WRK files. If the file is an SMF 0 file, where all channels appear on one track, the track is split so that each channel (0 to 15) is stored in the corresponding pattern, and pattern 16 contains the original track.

4.1.3 Menu / File / Open Playlist

The **Open Playlist...** menu entry opens a *Seq66* play-list file.

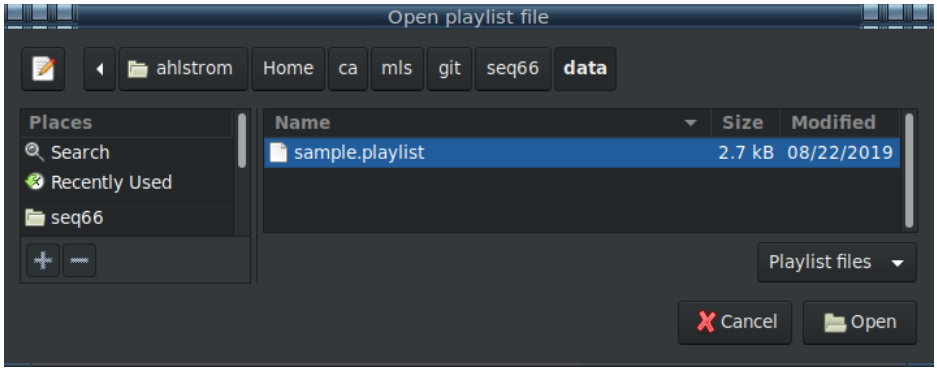


Figure 8: File / Open Playlist

The playlist file contains a list of "playlist sections", each listing a number of MIDI songs. These playlists and songs can be selected by the arrow keys or by MIDI control, and are displaed and editiable in the *Playlist* tab in the main window. See section 13 "Seq66 Play-Lists" on page 184.

4.1.4 Menu / File / Recent MIDI files

This menu entry provides a list of the last few MIDI files created or opened; play-list selections are *not* included in this list.

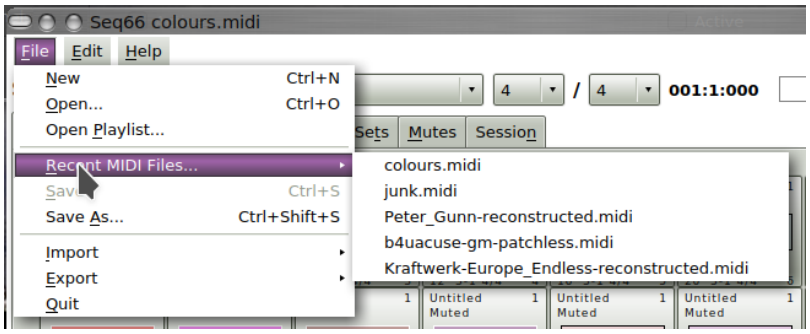


Figure 9: Seq66 Menu File Recent Files

Here is the long form when the 'rc' file's `[recent-files] full-paths` value is set to true:

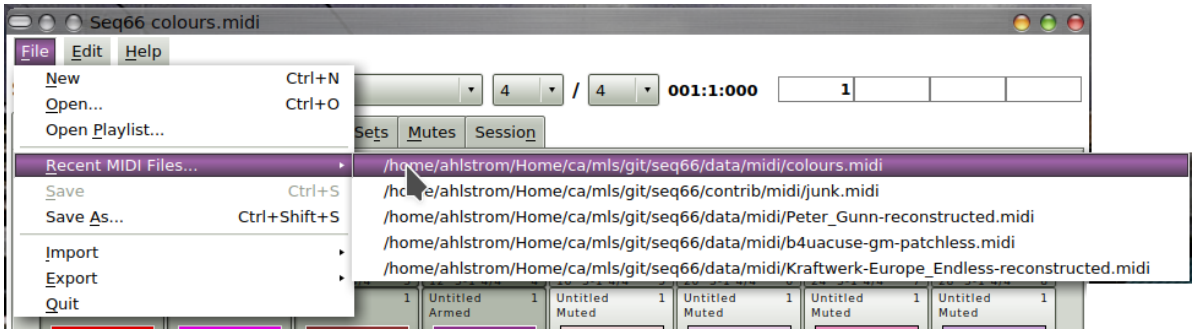


Figure 10: Seq66 Menu File Recent Files, Full Paths

This list is saved in the `[recent-files]` section of the 'rc' configuration file. In the 'rc' file, the full path to the file-name is stored. This path is in "UNIX" format, using the forward slash, or solidus, as the path separator, even in *Windows*. The `full-paths` option can be set to show the full path in the recent-files drop-down menu. Only unique entries are included in the recent-files list. The limit is 12 recent-file entries. This is a feature from *Kepler34* [8]. One can also set *Seq66* to load the most-recent file at startup. Here is an example from an 'rc' file:

```
[recent-files]
full-paths = false
load-most-recent = true
count = 3
/home/user/git/seq66/data/b4uacuse-gm-patchless.midi
/home/user/git/seq66/data/midi/colours.midi
/home/user/git/Julian-data/TestBeeps.midi
```

4.1.5 Menu / File / Save and Save As

The **Save** menu entry saves the song under its current file-name. If there is no current file-name, it opens up a standard file dialog to name and save the file. The **Save As** menu entry saves a song under a different name. It opens up the following standard file dialog, very similar to the **File Open** dialog, with an additional **Name** text-edit field.

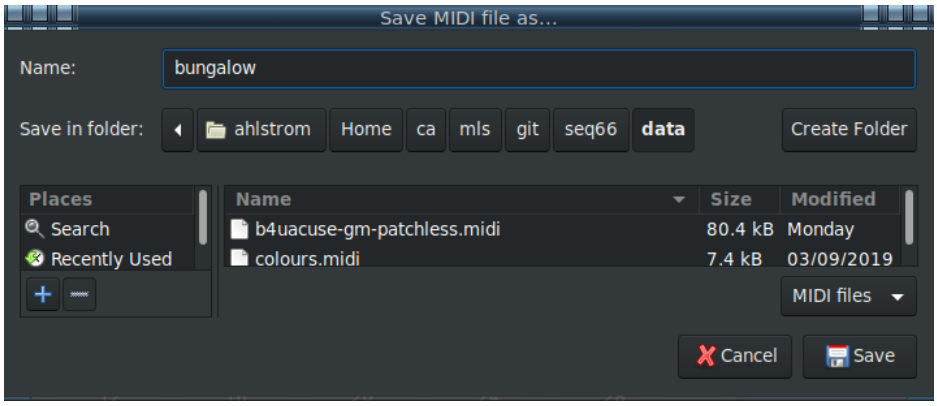


Figure 11: File / Save As

To save a new file or save the current file to a new name, enter the name in the name field, without an extension. *Seq66* will append a `.midi` extension to the filename. The file will be saved in a format that the Linux *file* command will tag as something like:

```
colours.midi: Standard MIDI data (format 1) using 16 tracks at 1/192
```

It looks like a simple MIDI file, and yet, if one re-opens it in *Seq66*, one sees that the mute-groups, labeling, pattern information, and song layout have been preserved in this file. This information is saved in a way that MIDI-compliant software should be able to use or ignore without failure. After the last track in the file, a number of sequencer-specific (SeqSpec) items are saved, to preserve the extra information that *Seq66* adds to the song. There is no way to save a *Cakewalk* "WRK" file.

Seq66 can only read them, and then save them as *Seq66* files.

Meta events are now partially handled by *Seq66*. Meta events **Set Tempo** and **Time Signature** are now fully supported. Other meta events, such as **Meta MIDI Channel** and **Meta MIDI Port** are now read as events, and are saved back when the file is saved. They cannot be edited in *Seq66*, but they are not lost. (Channel and port meta events are considered *obsolete* in the MIDI standard.) Lastly, various meta text events, such as **Lyric**, can be edited and saved.

4.1.6 Menu / File / Import / Import Project Configuration

This command is useful to grab an existing project configuration (i.e. the set of `qseq66.*` files) and copy it to another directory. This command is most useful in importing a project into a new NSM session. Previously, the "home" project would be imported automatically into a new NSM session, but this was deemed confusing by some users, and properly so!

This command brings up a file dialog box. Navigate to the desired source directory and then select the desired 'rc' file.

If importing into an NSM session, one must use the NSM user-interface (*agordejo*, *RaySession*, or *nsm-legacy-gui* to *Stop* and then *Start Seq66*. This saves and rereads the configuration. If importing into a normal (i.e. not NSM) session, *Seq66* will restart itself automatically to save and reread the configuration. Be aware!

Also note that importing a playlist is a separate operation (see below).

4.1.7 Menu / File / Import / Import MIDI to Current Set

The **Import** menu entry imports an SMF 0 or SMF 1 MIDI file as one or more patterns, one pattern per track, into the specified screen-set. This functionality is explained in detail in section 10.1.1 "Import MIDI to Current Set" on page 128.

4.1.8 Menu / File / Import / Import Playlist

A user can create a playlist that accesses MIDI files anywhere in the file system. However, in a session manager, it is preferable to have the configuration self-contained. Even without a session manager, it can be useful to copy a playlist to a subdirectory in order to separate it and its MIDI files from other playlists. Once a project has been imported or saved, then a playlist can also be imported, along with all of the MIDI files it references.

This command brings up a file dialog box. Navigate to the desired source directory and then select the desired 'playlist' file. This menu entry does a lot of things; here are the steps it performs:

- Copies the selected playlist file into the current configuration directory. This directory is one of the following:
  - The default "home" directory, `~/.config/seq66`.
  - A "home" directory specified by the `-home` option.

  - The session directory created by *NSM*.

  For discussion, this directory is called "HOME" or "home".
- In HOME, creates a directory called `playlist/listname`, where "listname" is the base part of the playlist name, as in `listname.playlist`.
- The playlist is opened, parsed, and all of the MIDI tunes specified in that file are copied into the new playlist directory, preserving the directory directory structure of the original playlist.
- The 'rc' file is modified to specify the new playlist, specify that it is `active`, to specify the new `base-directory` for all of the tunes, and to specify that the playlist will be saved at exit.
- Lastly, if not running within NSM, *Seq66* will be ***restarted*** automatically to load the new, active playlist. Within NSM, the user must stop and restart *Seq66* manually in the NSM user-interface..

4.1.9 Menu / File / Export / Export Project Configuration

This menu entry lets the user select a destination directory. Then the project files from the current "home" directory are copied to that destination directory. Useful for backup.

At the present time, any file specified in a play-list are *not* copied.

4.1.10 Menu / File / Export / Export Song as MIDI

Thanks to the *Seq32* project, the ability to export songs to MIDI format has been added. In this export, a complete song performance is recoded so that other MIDI sequencers can play the performance properly. This functionality is explained in detail in section 10.2.2 "Export Song" on page 130.

4.1.11 Menu / File / Export / Export MIDI Only

Sometimes it might be useful to export only the non-vendor-specific (non-SeqSpec) data from a *Seq66* song, in order to reduce the size of the file or to accomodate non-compliant sequencers. This functionality is explained in detail in section 10.2.3 "Export MIDI Only" on page 131.

4.1.12 Menu / File / Export / Export SMF 0

This feature is new since version 0.97. It allows all tracks in the song to be consolidated and exported in MIDI's SMF 0 format. It follows the same rules as song export. See section 10.2.4 "Export SMF 0" on page 132.

4.2 Menu / Edit

The **Edit** menu has undergone some expansion in *Seq66*.

1. **Preferences...**

2. **Song Editor**
3. **Apply Song Transpose**
4. **Clear Mute Groups**
5. **Reload Mute Groups**
6. **Mute All Tracks**
7. **Unute All Tracks**
8. **Toggle All Tracks**
9. **Copy Current Set**
10. **Paste To Current Set**

1. Preferences. This entry brings up a **Preferences** menu entry, to allow viewing and tweaking MIDI I/O ports, displays options, JACK options, and more. It can also be brought up by `Ctrl-P`. It is discussed in detail in a later section.

2. Song Editor. This item toggles the presence of the main song/performance editor. Note that the song editor is also available in the **Song** tab in the main window. The song/performance editor allows specifying exact numbers of loop replays; this provides a canned rendition of the MIDI tune.

3. Apply Song Transpose. Selecting this item applies the global song transposition value to all sequences / patterns marked as transposable. This actively changes the note / pitch value of all note and aftertouch events in the pattern. Normally, drum tracks are *not* transposable. For the setting of global song transpose, see section 7 "Song Editor" on page 100. Note that transpose can be enabled in the in the sequence editor (see section 6 "Pattern Editor" on page 74).

4. Clear Mute Groups. A feature of *Seq66* is that the mute groups are saved in both the 'rc' file *and* in the "MIDI" file. This menu entry clears them. If this resulted in any mute-group sequences status being set to false, then the user is prompted to save the MIDI file, so that it will no longer have any mute-group information. And then, if the application exits, the cleared mute-group information is also saved to the 'rc' file.

5. Reload Mute Groups. This menu entry reloads the mute-groups from the 'rc' file. So, if one loads a MIDI file that has its own mute groups that one does not like, this command will restore one's favorite mute-grouping from the 'rc' file.

6. Mute All Tracks. This menu entry, useful mostly in **Live** mode, immediately mutes *all* patterns in the entire song. The hard-wired keyboard short-cut for this action is `Ctrl-M`.

7. Unmute All Tracks. This menu entry, useful mostly in **Live** mode, immediately unmutes *all* patterns in the entire song. The hard-wired keyboard short-cut for this action is `Ctrl-U`.

8. Toggle All Tracks. This option toggles the mute/armed status of **all** tracks. It is useful mostly **Live** mode, which overrides **Song** mode even if the Song Editor is focussed. The hard-wired keyboard short-cut for this action is `Ctrl-T`.

9. Copy Current Set. This item marks the current set (i.e. the play-set) for the copying of all its patterns to another set. After clicking this menu entry, one can move to another set to paste it, using the following menu entry.

10. Paste To Current Set. Once a set has been copied into the internal set clipboard, then this menu item is enabled. Move to the desired set (whether empty or note), and then click this

menu item. All of the patterns in the original set are pasted into the current set, *overwriting all patterns* already in the set. Also note that the set clipboard can be pasted after a **File / New** or **File / Open**, to copy it to another file.

4.2.1 Menu / Edit / Preferences

Preferences provides a number of settings in one tabbed dialog, shown in the figures that follow. It allows one to set MIDI clocking, MIDI Input, display tweaks, minor playback options, and some JACK parameters.

Configuration items not (yet) implemented in **Preferences** are incoming MIDI events to control the sequencer; what keys are mapped to functions; how the mouse works, and a few other. The MIDI and Key controls, far more numerous than in *Seq24*, have been consolidated into a 'ctrl' file and are fairly easy to edit with a text editor. *Seq66* does not support the 'fruity' mouse mode at this time. If you want it, ask us!

4.2.1.1 Menu / Edit / Preferences / MIDI Clock

Note: *The MIDI Clock tab is sometimes difficult to click on.* We are not sure why. It seems to be theme-dependent. In some things the tab thumb changes color when it can work. Just keep clicking in various locations in the tab, or use the `Alt-C` key. Weird.

The **MIDI Clock** tab provides a way to set MIDI clocking for the available MIDI output busses. It configures the output busses for MIDI clock and data. It shows the devices that can play music. The items that appear in this tab depend on:

- What MIDI devices are connected to the computer. MIDI controllers, USB MIDI cables, applications with virtual ports, and other connected devices will add MIDI output devices (ports) to the system. This list will generally match the output of `aplaymidi -l` or `aconnect -lio`.
- The setting of the "manual-ports" option, which tells *Seq66* to set up virtual MIDI ports. It is enabled by the `--manual-ports` command-line option or the `[manual-ports]` section of the `qseq66.rc` configuration file, or in the **MIDI Input** tab described below.
- The setting of the *Seq66*-specific "reveal ALSA ports" option, `--reveal-ports` command-line option or the `[reveal-ports]` section of the `qseq66.rc` configuration file.

If `--manual-ports` is on, this list shows the virtual MIDI output busses that *Seq66* can drive. One needs to use a JACK or ALSA MIDI connection application to connect a device on each of those outputs. The fact that the the buss names can start with different numbers, depending on the system setup, can complicate the playing of MIDI in this manner. Also, the 'usr' configuration file can change the visible names of the ports to match specific equipment attached to the ports.

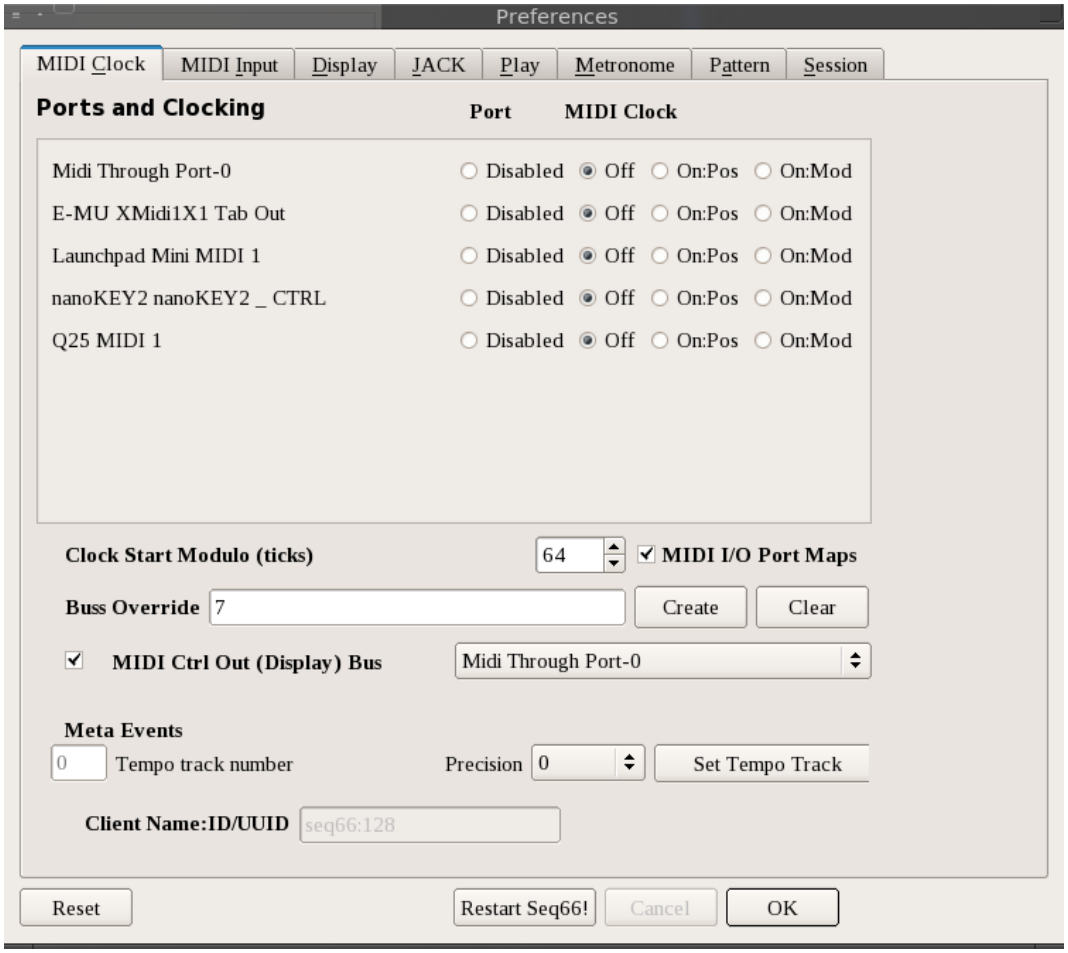


Figure 12: MIDI Clock (Output) Tab

This diagram is slightly out-of-date, but we go forward! It shows the tab for configuring MIDI output and clocking features.

> **Tip**: With some Qt themes, it is difficult to activate this tab by clicking because there is little or no sensitive area on the tab. In this case, click `Alt-c` once or twice.

Port-mapping is the default, and when active, is shown in this pane. If there are more than about a dozen output ports in the system, then a vertical scrollbar appears. The following elements are present in this tab:

1. **Ports and Clocking Table**
2. **Clock Start Modulo**
3. **Buss Override**
4. **MIDI I/O Maps**
5. **Create (maps)**
6. **Clear (maps)**
7. **MIDI Control Output Bus**
8. **Meta Events**
9. **Client Name:ID/UUID**

10. **Restart Seq66!**

1. Ports and Clocking.

This table shows the available MIDI outputs and their status. If the set of system MIDI devices and software devices has changed since the last run, this list could be in error. Restart the application and see if it is now correct. Currently, there is no way to edit the list except in the 'rc' file.

The **Ports and Clocks** table contains the following elements, although some can be removed by specifying the `port-naming = short` option in the 'rc' file.

1. **Index Number**
2. **Client Number**
3. **Port Number**
4. **Buss Name**
5. **Port Disabled**
6. **Off**
7. **On (Pos)**
8. **On (Mod)**
9. **Clock Start Modulo**

The format of the left side of the entry listing is like the following when the port-naming option is "long", and the MIDI subsystem is ALSA):

```
[5] 128:4 yoshimi:input
 ^  ^   ^ ^        ^
 |  |   | |        |
 |  |   | |         ---- Port/buss name
 |  |   |  ------------- Client name
 |  |    -------------- Port/buss number
 |   ------------------ Client number
  ---------------------- Index number
```

1. Index Number. The number in square brackets is an ordinal indicating the position of the output buss in the list. For all practical purposes in *Seq66*, it *is* the buss/port number. This number can be stored in a pattern in order to have the pattern's output go to that buss. This is true even if port-mapping is in place. It can be used with the `-b`, `--buss`, or `--bus` options to redirect all pattern output to that buss, useful if only one buss is active or the *Seq66* patterns route to non-existent busses. (See section 3.2.3 "Buss Override for Play-set" on page 22, and section 12.4.6 "'usr' File / User MIDI Settings" on page 160.)

2. Client Number. The number that precedes the colon is the "client number". It is useful mainly in ALSA, where clients can have numbers like "14", "128", "129", etc. For native JACK mode, it matches the index number or is the name of the client (e.g. "seq66").

3. Port Number. The number that follows the colon is the "port number". It is useful mainly in ALSA. For native JACK mode, it matches the index number.

4. Buss Name. These labels indicate the output busses (ports) available. *Seq66* does not access devices by name, but by port number. However, a port-map can be created to make it possible to find the correct buss / port number by name lookup.

5. Port Disabled. The **Port Disabled** clock choice marks an output port that the user does not want to use or that the operating system (*Windows* ☺) is locking or disabling. Normally, this inaccessible port would cause *Seq66* to exit. With the port disabled, the inaccessible port is ignored. This feature also shows when a port-map cannot find a device in the system's device list. When the *Windows* version of *Seq66* (`qpseq66.exe`) is first started, it may error out. It will then write a default `qseq66.rc` or `qpseq66.rc` configuration file, which can be examined to find the offending buss, which can then be marked in the normal 'rc' file as disabled.

6. Off. Disables the MIDI *clock* for the given output buss. MIDI output is still sent to those ports, and each port that has a device connected to it will play music. Some synthesizers may require this setting.

7. On (Pos). MIDI clock will be sent to this buss. MIDI Song Position and MIDI Continue will be sent if playback starts at greater than tick 0 in Song mode. Otherwise, MIDI Start will be sent. Note: In case of trouble, see section 19.3 "ALSA / Trouble-Shooting" on page 218.

8. On (Mod). MIDI clock will be sent to this buss. MIDI Start will be sent, and clocking will begin once the Song Position has reached the start modulo of the specified size (see the next item's description). This setting is used for gear that does not respond to Song Position.

Below the **Ports and Clocks Table** are more configuration elements.

1. Clock Start Modulo (ticks). This value starts at 1 and ranges up to 16384, and defaults to 64 ticks. It is used by the **On (Mod)** setting discussed above. It is the `[midi-clock-mod-ticks]` option in the *Seq66* 'rc' file.

2. MIDI I/O Port Maps. If checked (the default), then port-mapping is employed. This makes it a bit easer to manage MIDI devices across systems and to store the numbers in each pattern. Note that both input and output port mappings are activated by this checkbox. If changed, the **Restart Seq66!** button is enabled.

3. Create (maps). Pressing this button saves the current set of MIDI I/O ports to sections in the 'rc' file. These sections can be enabled in order to support port-mapping in subsequent runs of *Seq66*. Generally, after pressing this option, one will want to stop *Seq66*, rearrange the clock and input maps in the 'rc' file with a text editor, back up this file in a safe place, and restart *Seq66*.

4. Clear (maps). Pressing this button removes the port mapping. Once done, either restart *Seq66* or go to the **Session** tab and click the **Restart** button. (See section 24.1 "Concepts / Reload Session" on page 247.)

5. MIDI Control Output Bus. Use this control to select the output bus used to display application-automation status, loop status, and mute-group status. Requires a *reload session* to take effect. The number of the buss is stored in the 'ctrl' file named in section 4.2.1.10 "Menu / Edit / Preferences / Session" on page 53, as the value of `output-buss`. If port mapping is enabled (now the default), the nick-name of the bus is stored instead of the number.

6. Meta Events. This section consists of the following items:

1. **Tempo track number**
2. **BPM Precision**
3. **Set Tempo Track**

Tempo track number allows the user to move the tempo track from pattern 0 to another pattern. Changing this option is not recommended, since track 1 (0) is the official track for tempo events, but *Seq66* allows the user to record tempo events to another track. *Seq66* will process tempo events in any pattern.

Precision allows setting the number of digits past the decimal point to 0, 1, or 2. This is also a 'usr' setting. See section 12.4.6 "'usr' File / User MIDI Settings" on page 160. The BPM (tempo) is stored in the MIDI file multiplied by 1000 to accommodate the decimal places.

Set Tempo Track Enabled when a valid tempo track number is given. It makes the tempo track official if it is not zero anymore.

7. Client Name:ID/UUID. This read-only text field shows two things:

1. **Client Name**. This is the name of the client under ALSA or JACK. It defaults to `seq66`, but it can be altered by the command-line option `--client-name` or by a session manager. Each instance of Seq66 run under ALSA will have a different client ID.
2. **ID/UUID**. Under ALSA, the client number (client ID) is shown. Under JACK, the UUID that JACK assigned to *Seq66* is shown.

8. Restart Seq66!. Certain changes require a *Seq66* restart, unfortunately. When enabled, clicking this button does not exit *Seq66*, but it does cause all of the internal mechanisms to be recreated from scratch.

4.2.1.2 Menu / Edit / Preferences / MIDI Input

To set up *Seq66* to record MIDI from devices such as controllers and keyboards, the output of the ALSA MIDI recording command-line `arecordmidi -l` is relevant. Something like that listing appears in the Input tab:

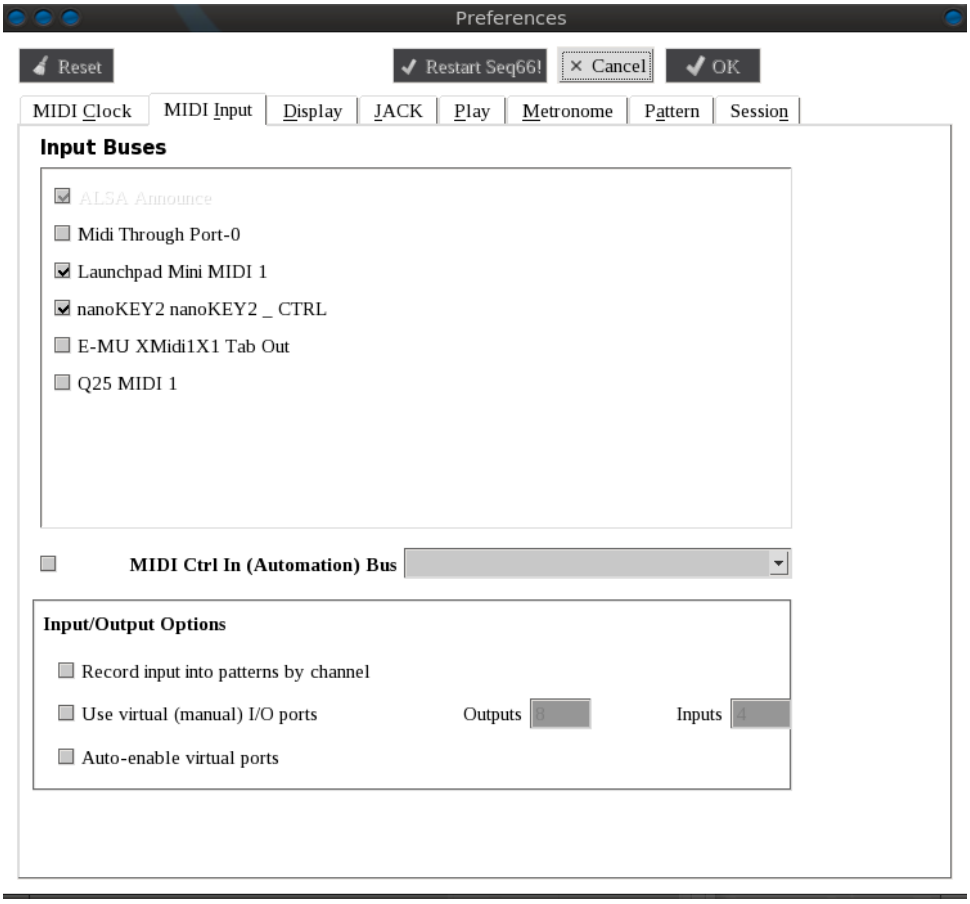


Figure 13: MIDI Input Tab

Port-mapping is the default, and when active, is shown in this pane. If there are more than about a dozen input ports in the system, then a vertical scrollbar appears.

Any item checked allows *Seq66* to record MIDI from that source, which must be connected to this input port.

Warning: If the `[user-midi-bus-definitions]` value in the 'usr' configuration file is non-zero, and the corresponding number of `[user-midi-bus-N]` settings are provided, then the list of existing hardware will be ignored, and those values will be shown instead. This feature can be overridden with the `--reveal-ports` (`-r`) option. If you define these sections, they should match your hardware exactly, and your hardware should not change from session to session (or port-mapping should be enabled). If the "auto ALSA ports" option is turned on, via the `-a` or `--auto-ports` option, then the input ports from the system are shown.

1. **Input Buses.** **Input Buses** delineates the MIDI input devices as noted above.

2. **MIDI Control Input Bus.** Use this control to select the input bus used for MIDI control automation of application actions, loop actions, and mute-group actions. Requires a *reload session* to take effect. The number of the buss is stored in the 'ctrl' file named in section 4.2.1.10 "Menu / Edit / Preferences / Session" on page 53, as the value of `control-buss`. If port mapping is enabled (now the default), the nick-name of the bus is stored instead of the number.

3. Input Options. **Input Options** adds further refinements to MIDI input. Currenty it has only one setting, for recording input into patterns by the channel in each event.

4. Record-by-Bus. **Record into patterns by buss** causes MIDI input from multiple busses to be distributed to each sequence according to MIDI input buss number.

5. Record-by-Channel. **Record into patterns by channel** causes MIDI input with multiple channels to be distributed to each sequence according to MIDI output channel number.

Only one of these record-by options can be enabled at the same time. The record-by-buss option takes precedence.

When these options are disabled, the normal recording behavior dumps all data into the current sequence, regardless of channel or buss. See section 11 "Seq66 Recording In Depth" on page 132, which describes recording in more detail.

6. Input/Output Virtual Ports. **Use virtual (manual) I/O ports** This option allows for configuration of the manual-ports option from within the user-interace. Once the option is enable A *reload session* (see section 24.1 "Concepts / Reload Session" on page 247) is necessary for this option to take effect.

7. Virtual Ports Auto-Enable. **Auto-enable virtual I/O ports** If set, the ports are all automatically enabled upon a restart. The following figure shows that a large number of virtual ports can be defined.

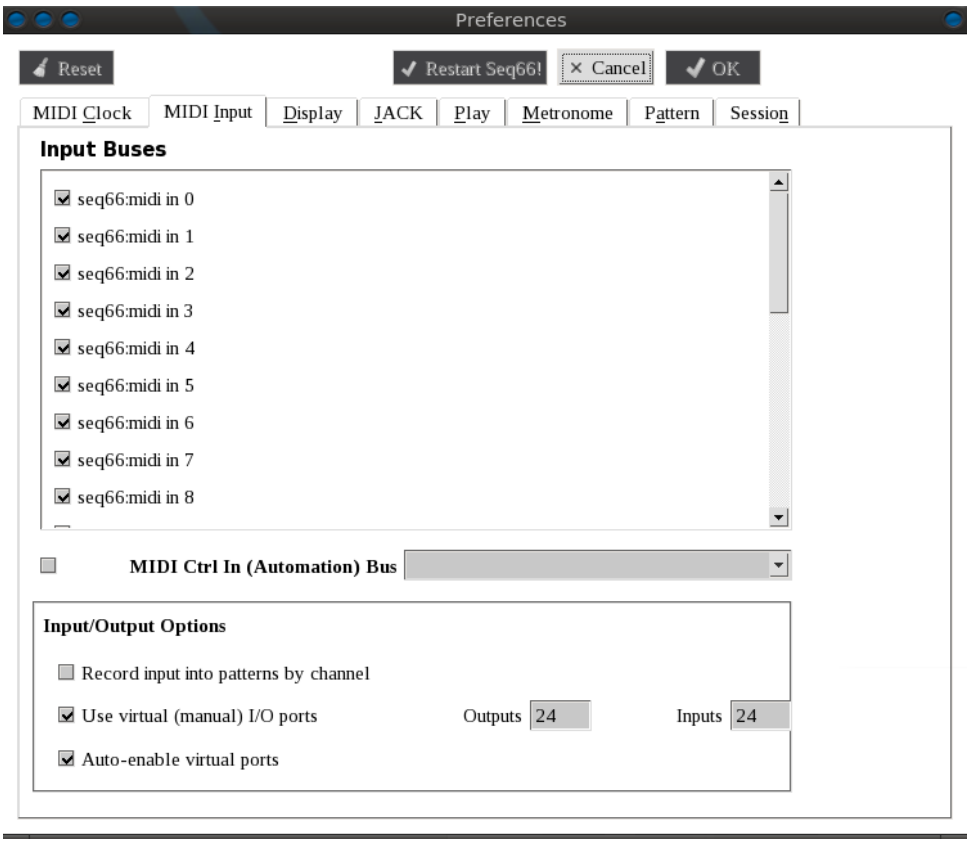


Figure 14: MIDI Virtual Inputs

Note that the user is responsible for connecting the virtual MIDI ports, using something like *aconnect* (ALSA) or *qjackctl* (JACK).

4.2.1.3 Menu / Edit / Preferences / Keyboard (removed)

Unlike *Seq24*, *Seq66 does not* provide an options tab for setting up the keyboard. There are just too many new keystroke-automation functions to fit in a configuration dialog box. The default keyboard mappings follow *Seq24* fairly well, but add a large number of additional controls; around 96 keystroke slots would need to be provided! The keystroke and MIDI controls are consolidated, and are easy to change by editing the appropriate 'ctrl' configuration file, stored in one of the following directories, depending on the operating system:

```
/home/username/.config/seq66/qseq66.ctrl           (Linux)
C:/Users/username/AppData/Local/seq66/qpseq66.ctrl (Windows)
```

There are also some extended examples present in the *Seq66* `data/linux` and `data/samples` directory. Also see section 23 "Launchpad Mini" on page 236. For more information on keystrokes, see section 17.1 "Keyboard Control" on page 202.

4.2.1.4 Menu / Edit / Preferences / Mouse (removed)

Unlike *Seq24*, *Seq66 does not* provide an options tab for the mouse-interaction method. It is not supported in *Seq66*... the **Fruity** interaction method is not available; only the **Seq24** interaction is available.

4.2.1.5 Menu / Edit / Preferences / Display

This dialog provides a few odds and ends to enhance the user-interface. Some of these items (plus a few more) can be configured by editing the 'usr' file.

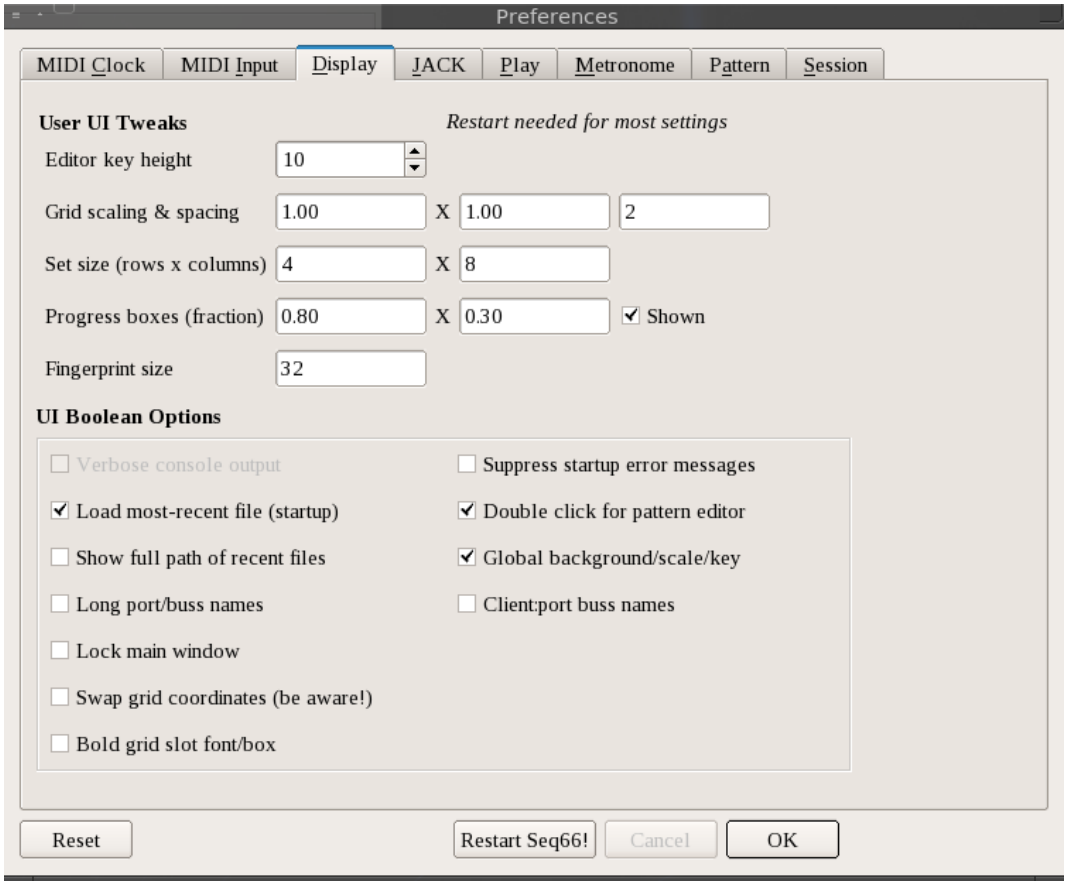


Figure 15: Display Options

1. Editor Key Height. This option affects the pattern editor's piano roll. Smaller means a wider range of notes can be shown. There are also **-**, **0**, and + buttons in the pattern editor that provide vertical zoom.

2. Grid scaling & spacing. These three items set scale factor for width and height of the main window, and adjust the spacing between the grid slots.. The lowest scale factor is 0.5, and the largest scale factor is 3.0. For the smallest window, the smallest practical values are 0.85 x 0.60. The spacing unit is pixels.

3. . Provides a way to change the set size. The default is **4 x 8** (rows by columns), but we intend to support **4 x 4**, **8 x 8**, and **12 x 8** as well.

Warning: A different set size alters the 'ctrl' file layout radically. We still have to work through all the implications of changing the set size, so back up your configuration and proceed with caution!

4. Progress Boxes. Provides a way to change the size of the progress box in each button. Values are width and height fractions (up to 1.0) re the button size. This is a 'usr' option.

5. Progress Box Shown. If the **Shown** check-box is *unchecked*, then the progress boxes and pattern color are not shown. This is a 'usr' option.

6. Fingerprint Size. This value, if set from 32 to 128, indicates the number of events above

which a "fingerprint", rather than every note, will be drawn. It can save some CPU time in drawing the grid. If set to 0, the whole pattern is drawn, no matter how long the pattern is.

7. Verbose Console Output. This boolean makes more output appear if *Seq66* is run from a console/terminal. It will also increase the amount of data logged to the log file, if activated. It is only a temporary setting, just like its command-line counterpart, `-verbose`; when *Seq66* exits, the setting remains false.

8. Suppress startup error messages. Unlike the verbose setting, this one is sticky. It prevents the display of error prompts at startup. It is useful when the system keeps flagging the same problem, it cannot be fixed, and can be ignored. It is *not* the opposite of "verbose".

9. Load Most Recent File (startup). If checked, the file at the top of the `[recent-files]` list in the 'rc' file is loaded at startup.

10. Show Full Path of Recent Files in Menu. The full path of each file in the `[recent-files]` list is shown in the menu. Although they can be uncomfortably long, they can show files that have the same name, but in different directores.

11. Long Port/Buss Name. Controls how much port information is shown in the clocks and input listings. For the "portmidi" (e.g. *Windows*) implementation, keep this option checked.

12. Lock Main Window. This item makes the window non-resizable after startup.

13. Swap Coordinates. Normally, *Seq66* displays the pattern and mute-groups grids where the pattern numbers increase fastest downward. Some might prefer to have pattern numbers increase fastest rightward. This setting make the patterns show in the more conventional manner.

Warning:

- This setting requires the 'ctrl' file to be rewritten if one want to preserve the normal layout for the pattern hot-keys and the mute-group hot-keys.
- This setting has not been rigorously tested, so be prepared for some issues.

A 'ctrl' file for the swapped setting is provided in `qseq66-swapped.ctrl` in the `data/linux` directory, but it might not be completely correct yet.

14. Bold Slot Font. This setting makes the font in the live grid bold, and it allows make the progress-bar thick in the grid and in the Live and Song piano rolls. It is the same as the `progress-bar-thick = true` option in the 'usr' file. See section 12.4.3 "'usr' File / User Interface Settings" on page 158.

15. Double click for pattern editor. If set, a double-click on a grid button brings up the pattern for editing. Disable it if the effect is confusing.

16. Global background/scale/key. If set, setting the background sequence, scale to show, or the key of the track will apply to all pattern windows that are opened.

17. Client:port buss names. If checked the MIDI engine's "client:port" numbers are shown in the port listings.

4.2.1.6 Menu / Edit / Preferences / JACK

This tab sets up JACK transport, if *Seq66* was built with JACK support (*Linux* only).

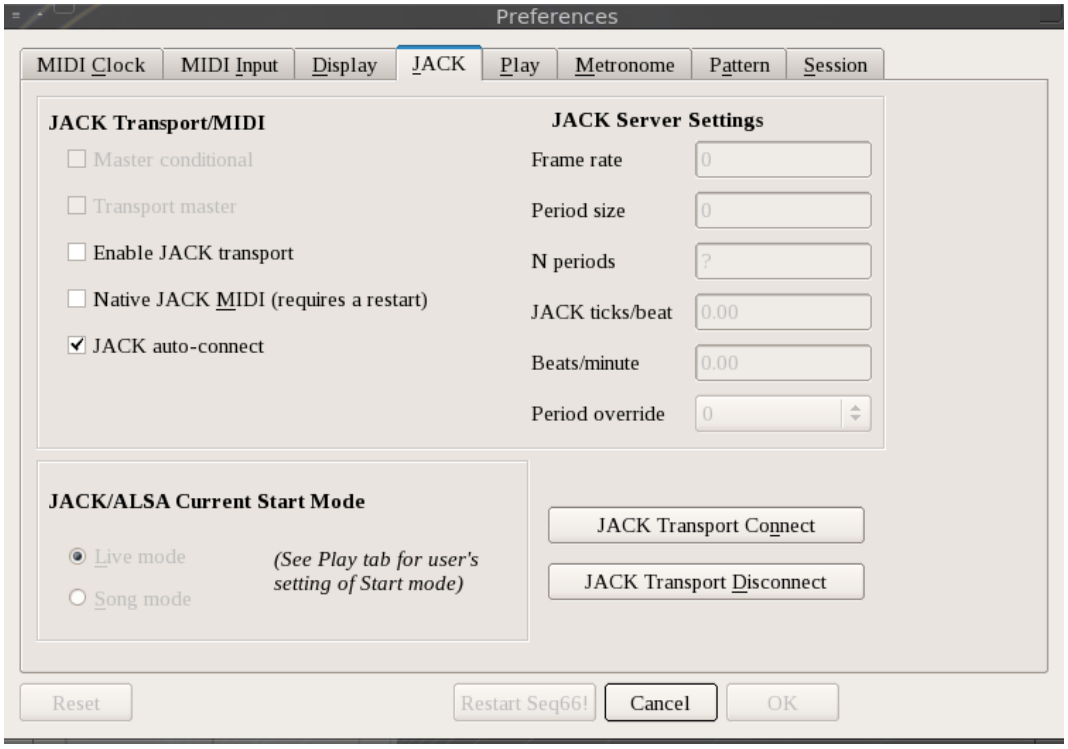


Figure 16: Edit / Preferences / JACK

The main sections in this dialog are:

1. **JACK Transport/MIDI**
2. **JACK Start Mode**
3. **JACK Transport Connect and Disconnect**
4. **JACK Server Settings**

1. Transport/MIDI. These settings are stored in the 'rc' file settings group `[jack-transport]`. This items collects the following settings:

- **Jack Transport**. Enables slave synchronization with JACK Transport. The command-line option is `--jack-transport`. The behavior of this mode of operation is perhaps not quite correct. Even as a slave, *Seq66* can start and stop playback. Note that this option cannot be disabled via the mouse if the **Transport Master** option is enabled. Disable that one first.
- **Transport Master**. *Seq66* will attempt to serve as the JACK Master. The command-line option is `--jack-master`. If this option is enabled the **JACK Transport** option is automatically enabled as well.
- **Master Conditional**. *Seq66* will fail to serve as the JACK Master if there is already a Master. The command-line option is `--jack-master-cond`. If this option is enabled the **JACK Transport** option is automatically enabled as well.

- **Native JACK MIDI**. This option is for the `qseq66` (Linux) version of *Seq66*. If set, MIDI input and output use native JACK MIDI, rather than ALSA. However, if JACK is not running on the system, then `seq66` will fall back to ALSA mode. (However, if `jackdbus` is running, but the JACK engine is not, then a couple of non-working manual ports are created. To be fixed in the future.) The command-line option is `--jack-midi` or `--jack`.
- **JACK Auto-Connect**. This option has been true for a long time in *Seq66*, and non-configurable. Now it can be turned off, in order to let the user or a session manager make the connections, even when not using manual/virtual ports.

  Tip: Seq66 generally works better as JACK Master than JACK Slave.

If one makes a change in the JACK transport settings, it is best to then press the **JACK Transport Disconnect** button, then the **JACK Transport Connect** button. Another option is to restart *Seq66*... the settings are automatically saved when *Seq66* exits.

2. JACK Start mode. This item collects the following settings, also stored in the 'rc' file settings group `[jack-transport]`.

- **Live Mode**. Playback will be in live mode. Use this option to allow muting and unmuting of patterns. This option might also be called "non-song mode". The command-line option is `--jack-start-mode 0`.
- **Song Mode**. Playback will use only the Song Editor's data. The command-line option is `--jack-start-mode 1`.

Seq66 also selects the playback modes according to which window started the playback. *The main window*, or pattern window, causes playback to be in live mode. The user can arm and mute patterns in the main window by clicking on sequences, using their hot-keys, and by using the group-mode and learn-mode features. The song editor causes playback to be in performance mode, also known as "playback mode", or **Song** mode.

3. Connect. Connect to JACK Sync. This button is useful to restart JACK sync when making changes to it, or when *Seq66* was started in ALSA mode.

4. Disconnect. Disconnect from JACK Sync. This button is useful to stop JACK sync when making changes to it. JACK connection and disconnection are disabled during playback, but the buttons don't yet reflect that status.

5. JACK Server Settings. This read-only section shows the current settings of the JACK server, as much as possible.

4.2.1.7 Menu / Edit / Preferences / Play Options

This tab contains some disparate options ostensibly related to playback.

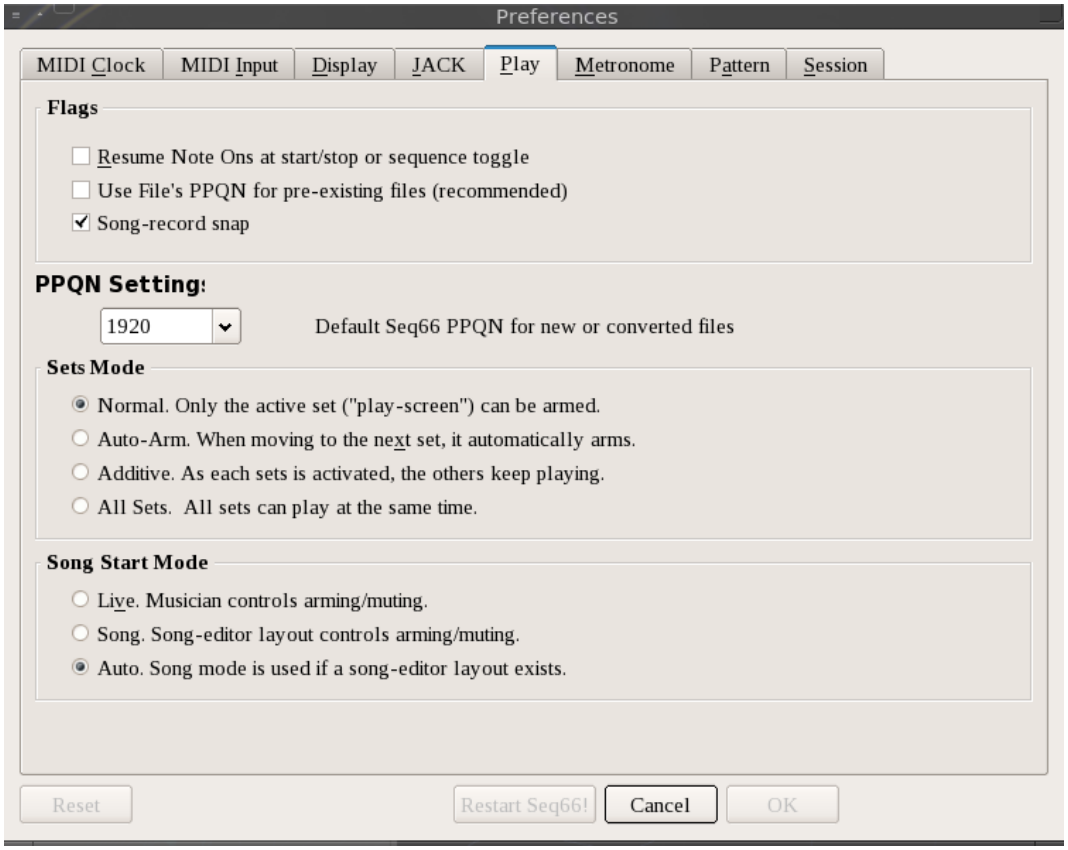


Figure 17: Play Options

1. Resume Note Ons.... Resume Note Ons at start/stop or sequence toggle allows notes that had already started to be resumed when playback resumes.

2. Use File's PPQN.... Use File's PPQN for Pre-Existing Files, if checked, allows *Seq66* to run using the PPQn of the MIDI file rather than the default *Seq66* internal PPQN. When this option is changed, the `--user-save` option is turned on to preserve the setting when *Seq66* exits.

3. Default Seq66 PPQN.... Default Seq66 PPQN for New or Converted Files, if checked, allows the standard PPQN, 192 pulses/quarter-note, to be changed to discrete values from 32 to 19200. Intermediate values, even oddball values, can be entered by typing the number directly. When this option is changed, the `--user-save` option is turned on to preserve the setting when *Seq66* exits. If there is a MIDI file loaded, it is modified to use the new PPQN, and the user is prompted to save it at exit.

4. Sets Mode. This item determines how sets are handled. Recall that a set is a number of patterns (up to 4x8) in the pattern grid, and that the current set is the one visible in the pattern grid. The way sets work in *Seq66* is that, when a set is selected, all the patterns in it are loaded into what is called the "play-set". When play starts only, patterns in the play-set are handled. The **Sets Mode** option allows special handling of the play-set.

1. **Normal**. In this mode, only the current set's patterns can be unmuted. When switching to another set, the current set's patterns become muted, and the new set's patterns are shown,

unmuted.
2. **Auto-Arm**. Here, when the new set is loaded, it is immediately unmuted.
3. **Additive**. With this option, when a new set is loaded, the previous set keeps playing. This allows a build-up of patterns in playback.
4. **All Sets**. Here, all sets in the tune are loaded and unmuted at once. Try this mode with the `b4uacuse-stress.midi` file in the *Sequencer64* project. It's a good test of *Seq66* and your hardware/software synthesizer!

One can clear the out play-set, and set only the current set active, by clicking the exclamation point button to the left of the "Active" label at the bottom of the main windows.

4.2.1.8 Menu / Edit / Preferences / Metronome Options

This tab contains options for the "metronome" and "background recording" features:

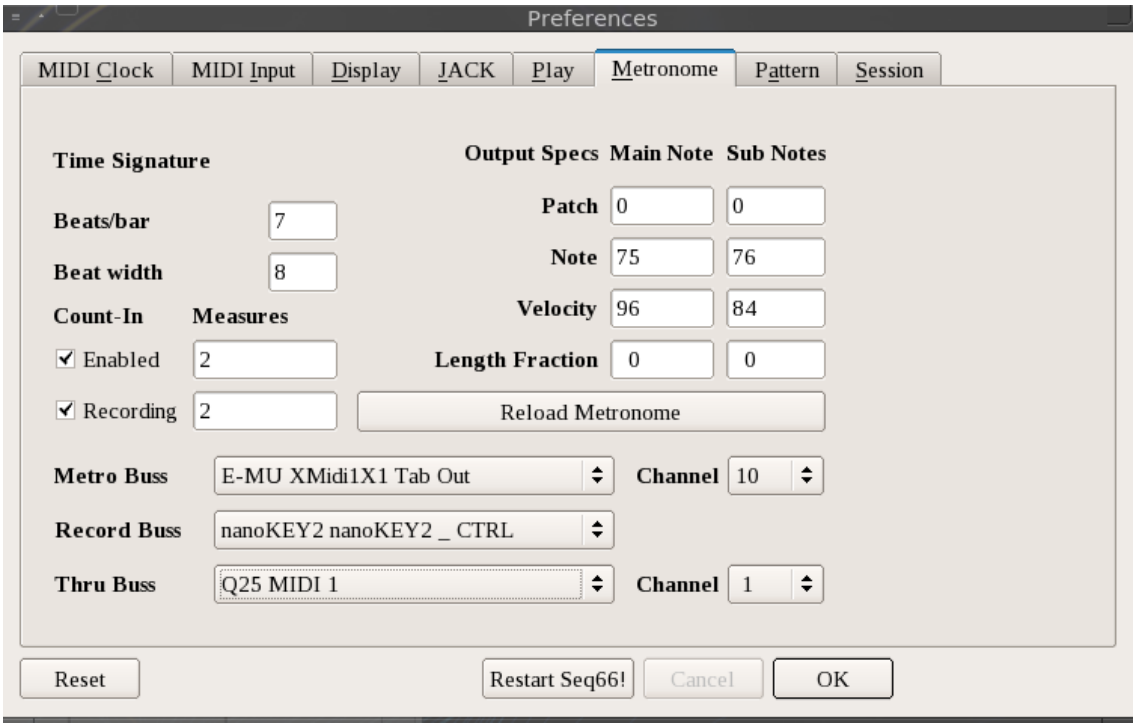


Figure 18: Metronome Options

The metronome feature is enabled in the main live grid via a metronome button. The metronome is a standard *Seq66* pattern that is used for playback of the metronome, but it is never seen nor directly edited by the user. It is not saved with a song, so changing the metronome does not modify the song. The settings shown above are saved to a "metronome" section in the 'rc' file. The metronome is a pattern that first plays a main note once, and then plays "sub" notes for the rest of the measure. Here are the settings:

1. Beats/bar. This setting sets the beats-per-measure for the metronome only. It currently does not affect the time-bar in the main window. Should it? There is a global beats/bar as well as

beats/bar for each pattern.

2. Beat width. This setting sets the beat width for the metronome only. The following settings are provided for the "main" note (the note that occurs on the beginning of the measure) and the "sub" notes (the notes that occur on each beat):

3. Patch. This item sets the program (patch) number for the note, which sets the instrument to play for the notes. We currently do not have a drop-down box to select the patch by name. The default patch is 0. As noted below, the default channel is 10, so this patch is the "Standard Drum Kit" for the device. Thus, by default the metronome can be implemented by two different drums.

4. Note. This item provides the note value to be played. Recall that 60 is the same as "middle C". By default, the main note is 75, the "Clave" for the drum kit, and the sub note is 76, the "High Wood Block" for the drum kit.

5. Velocity. This item provides the note velocity to be played, to provide an accent on the main note.

6. Length Fraction. The length of the notes are specified as a fraction of the beat width, and this value ranges from 0.125 to 1.0 to 2.0. If set to 0, the length is half of the beat width.

7. Reload Metronome. This button pauses playback (if playing), loads in the new metronome settings, and continues playing (if it was playing). It is *not* enabled when the status/configuration of background recording changes.

8. Metro Buss. This value selects the output MIDI device to use to play the metronome. It *must* be enabled in the **MIDI Clock** list.

9. Channel. This value selects the channel to use to play the metronome.

10. Record Buss. This value selects the input device to use to record events into the background pattern. Note that this device *must be enabled* in the **MIDI Input** buss list.

11. Thru Buss. This value selects the output MIDI device to use to play the incoming background record notes. Otherwise they will not be heard. It *must* be enabled in the **MIDI Clock** list.

12. Thru Channel. This value selects the channel to use to play the recorded notes as they come in.

We still have some more work to do to refine the metronome, the background recorder, and their configuration, pending user input. For more information about the metronome, see section 5.1.3.3 "Metronome" on page 69. For information on count-in background recording, see section 5.1.3.4 "Count-in Recording" on page 70.

4.2.1.9 Menu / Edit / Preferences / Pattern

This tab provides options for the status of newly-created patterns and for randomization of amplitude and jitter of time-stamps.

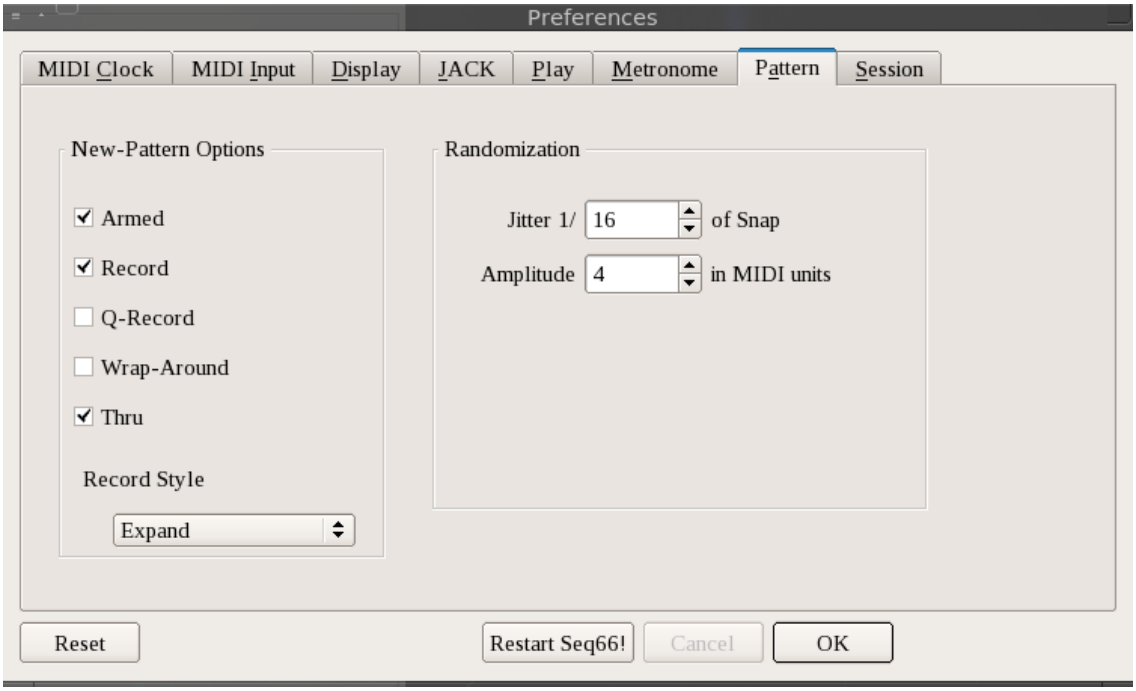


Figure 19: Pattern Options

13. Pattern. **Pattern** This relatively new tab provides a way to configure the status of a newly-created pattern. It also provides a way to change the range of amplitude randomization and time jittering of events.

The first section is **New-Pattern Options**. It defines the statuses of a newly-created pattern. This can be convenient for live-recording. The status setting are:

- **Armed**. This setting causes the pattern to be armed when a new pattern is created.
- **Record**. The new pattern starts in record mode.
- **Tighten Record**. The new pattern starts in tightened (partly quantized) record mode.
- **Quantize Record**. The new pattern starts in quantized record mode.
- **Note-map Record**. The new pattern starts in note-mapping record mode. Notes are translated live via a 'drums' file, if set active in the 'rc' file.
- **Wrap-Aound**. The new pattern will allow prolonged notes to wrap around so that the Note Off event precedes the Note On event in the pattern loop.
- **Thru**. The new pattern starts with MIDI Thru enabled.
- **Record Style** This setting sets how record mode works for the pattern.
  - **Merge**. As recording and looping proceeds, new events merge with the existing events.
  - **Overwrite**. When the pattern loops back to its beginning, any existing events are deleted. A good way to try to get the right collection of notes.
  - **Expand**. When recording as notes are recorded, the pattern expands to accomodate them. This results in a longer pattern than initially specified.
  - **Oneshot**. Events are entered until the end is reached. Useful for recording stock patterns from a drum machine.
  - **Oneshot Reset**. At the end of the specified length of the pattern, all events are cleared.

Normal recording is set. Need to look into this as we cannot rememember all the details :-D.

The second section is **Randomization**. The range of randomization is based on a range parameter, and goes from -range to +range. The concept of jitter means that the time-stamps of recorded events are randomized slightly. The concept of randomization means that the amplitudes of events are randomized slightly. The randomization settings are:

- **Jitter**. This value is a jitter divisor. It sets the fraction of of the current snap value that is used as the range of jittering the time. For example, "8" means that the range is 1/8th of the snap value.
- **Amplitude**. This value is used for various data values. For Notes On (but not Notes Off), this parameter affects the range of amplitude variation, when amplitudes are the standard MIDI range, 0 to 127.

One minor issue, which we're still trying to work around, is that our various randomization algorithms seem biased to emit negative numbers. If one clicks in a pattern editor piano roll, types `Ctrl-A` to select all notes (and aftertouch) and types the `r` key repeatedly to randomize note amplitudes, the overall velocities slowly descend to 0. Still not sure what's wrong with *Seq66* randomization, but it is only important if randomizing a large number of times.

The second section, not shown, is **Additional 'usr' Options**. It contains only one option at present, **Escape Can Close Pattern Editor**. If set (the default is false), then the `Esc` key can not only stop playing and exit paint mode, but can also close the pattern window.

4.2.1.10 Menu / Edit / Preferences / Session

This tab contains options related to session management and the configuration files.

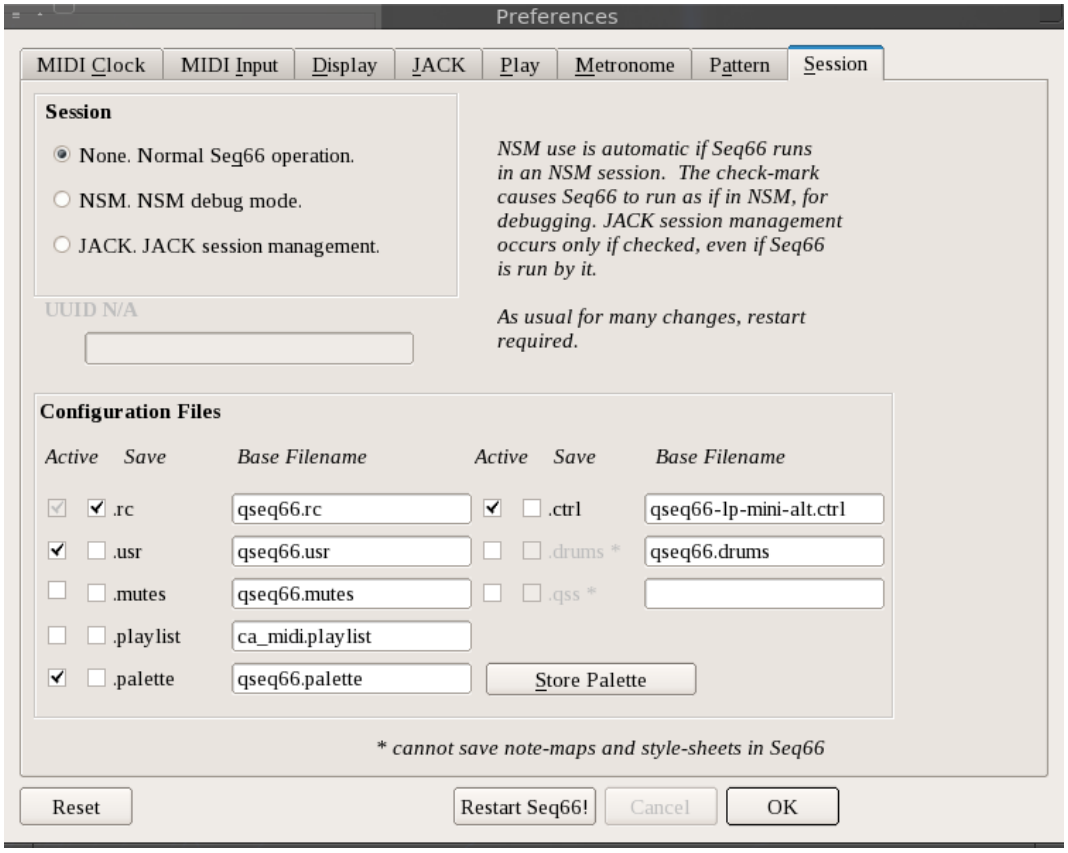


Figure 20: Session Options

1. Session. **Session** This tab provides for three modes of session management: None, the Non/New Session Manager (NSM), and JACK Session management. None is the normal mode of operation, where the user has full control of where to put files, what other applications are to be run alongside *Seq66*, and what connections are to be made.

NSM provides a rigorously-controlled session management, and directs *Seq66* what menu items to display, whether to hide the user-interface or not, where configuration files and MIDI files go, and what applications are run in a session. It can also (via `jackpatch`) keep a record of connections to reconstruct.

JACK Session provides a location for file and a record of applications and connections, but otherwise lets the user mess things up. It is provided because some people still use it. For more information about session management, see section 9 "Session Management" on page 116.

2. UUID. **UUID** is a read-only field that shows any UUID that's relevant to a session. Normally has a value only in a *JACK* or *NSM* session. Also see the **Session** tab in the main window (section 9 "Session Management" on page 116).

3. Configuration Files. **Configuration Files** shows the status of the configuration files. (See section 12.3 "'rc' File" on page 141). The 'rc' file is always active, and normally is saved at exit (even if no configuration changes occurred).

The 'usr' file should also be active, but one can disable it, which is currently an *experimental* and

untested option. Normally, it is not saved at application exit (except after the first run on one's system). (See section 12.4 "'usr' File" on page 153).

The rest of the configuration files are optional. See section 12.5 "'ctrl' File" on page 163, section 12.6 "'mutes' File" on page 181, section 12.7 "'drums' File" on page 184, and section 13 "Seq66 Play-Lists" on page 184.

4. Palette File Base Name. This text edit holds the base name of a 'palette' file, which is always stored in the *Seq66* configuration directory. (See section 16 "Palettes for Coloring" on page 199.)

5. Save Current Palette. Normally, there is no palette file. Pushing this button creates one, which can then be modified and configured as the palette-file to use in the 'rc' file.

4.3 Menu / Help

The usual **Help** dialog is provided. As of version 0.98.8, it has been beefed up with a way to access a tutorial and the user manual.

These new help items are a work in progress, so please apprise us of any issues; include information on the operating system and, if *Linux*, the desktop/window manager in use.

4.3.1 Menu / Help / About...

This menu entry shows the "About" dialog. That dialog provides access to some credits for the program as well. authors and the project documentors, and active link to them. It also shows Git version-control information as well.

4.3.2 Menu / Help / Build Info...

This menu entry shows the "Build Info" dialog. This list of build options enabled in the current application is the same list that it generated via this command line:

```
$ seq66 --version
```

4.3.3 Menu / Help / Song Summary File...

This menu entry allows one to write a summary of the song data into a text file. It brings up a file dialog which defaults to the name of the currently-loaded MIDI file, with the extenstion `.text` and the directory from where the MIDI file was loaded. It shows the filename, the information about the sets and tracks, MIDI format (0 or 1), and the PPQN.

It also shows each sequence: name, channel (128 mean there is no output channel), the time signature, buss number (and any mapping), the length in pulses, the event and trigger count, transposability, key and scale, and color number (if any). For each trigger in the pattern, its start, stop,

offset, and transposition values are shown. This file can be helpful for trouble-shooting or solving puzzling effects in the tune.

4.3.4 Menu / Help / App Keys

This entry brings up a dialog that shows brief descriptions of the non-automation keys available in various contexts. These keys are almost exclusively hardwired and currently cannot be changed via a configuration file. By pressing a button, the desired keystrokes can be quickly viewed. Note that the descriptions come from small HTML files that are part of the installation.

4.3.5 Menu / Help / Tutorial

This entry brings up a short tutorial of *Seq66* in the default browser. This tutorial is meant only to jump-start a new user of *Seq66*, and is a work in progress. It does not cover nearly as much as the user manual, so check that out in the next section.

Normally, the tutorial will open a web page. If it does not, one might need to set up a default browser. On Linux, make sure that there is a "desktop" file for the browser, as in `/usr/share/applications/firefox.deskto` If so, then run the following command, and then test it:

```
$ xdg-settings set default-web-browser firefox.desktop
$ xdg-open https://ahlstromcj.github.io/docs/seq66/tutorial/index.html
```

On Windows, this procedure is still *to be determined*.

In both systems, one can override the default applications by opening the specified 'usr' file (usually `qseq66.usr` or `qpseq66.usr` and specifying the full path to the desired applications (Linux paths shown here):

```
[user-options]
log = "/home/user/.config/seq66/seq66.log"
pdf-viewer = "/usr/bin/zathura"
browser = "/usr/bin/google-chrome"
```

Also see section 12.4.7 "'usr' File / User Options" on page 161.

4.3.6 Menu / Help / User Manual

This menu entry first tries to locate the user manual on the internet and open it in the default browser. If not found, or the network is down, then this entry brings up the full *Seq66* user manual in the default PDF viewer. It currently looks in the possible installation areas and in the *Seq66* source tree to find the PDF.

On Linux, one can follow the setup procedure in the previous section and test it via the following command, which will show the manual in the default browser.:

```
$ xdg-open https://ahlstromcj.github.io/docs/seq66/seq66-user-manual.pdf
```

5 Patterns Panel

The main window of *Seq66* is where one creates a set of patterns (a "screen-set"), manages the configuration, controls the playback rate, adds tempo events, and opens the pattern, song, event, mute-groups, or playlist editors. See section 3 "Live Grid / Main Window and Tabs" on page 17. The *Seq66* menu bar is discussed in section 4 "Menu" on page 29. Here, we describe the patterns panel.

The **Patterns Panel**, also called the **Live Frame** or **Live Grid**, is the center of the **main window** of *Seq66*. See Figure 1 "Seq66 Main Screen, Annotated" on page 18.

In addition, an external MIDI file can be dragged onto this grid, and dropped, to easily open a MIDI file:

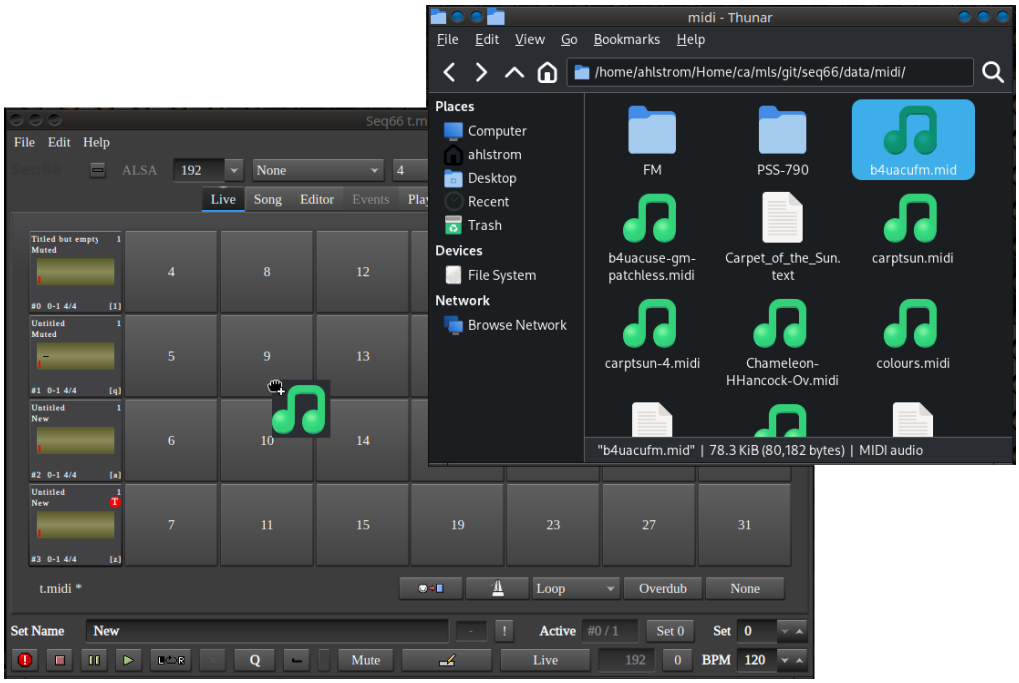


Figure 21: Patterns Panel Drag-and-Drop

If a modified file is loaded, the user will be prompted to save, discard, or cancel.

Seq66 works with "patterns" (also known as "loops", "tracks", or "sequences") that are repeated throughout a song. One composes and edits small patterns in a grid, and combines them to create a full song. This is a powerful way to work, and makes one productive quickly.

The musician can control the playback and muting/unmuting of each pattern in the song, while it is playing, from within this window. One can also switch to other screensets, to work with a different section of the song.

5.1 Patterns / Main Panel

The main panel of the application provides a grid of empty boxes, or boxes that have patterns, as shown in Figure 23 "Patterns Panel Pop-up Menu" on page 61. These boxes are referred to as "slots", and are implemented by push-buttons. Each filled slot represents a loop, track, sequence, or pattern (interchangeable terms). One sees only 32 loops at a time in the main panel, but many more than 32 loops are supported by *Seq66*.

This group of 32 loops is called a "screen-set". One can switch between sets by using the "[" and "]" keys on the keyboard, or by using the spin-widget labeled **Set**, or by hitting the (default) `Home` key to make it the playing screenset. There are a total of 32 sets, for a total of 1024 loops/patterns. Only one screen-set can be controlled at a time, in general. Multiple screensets can be playing at the same time, in some configurations.

The `Page Up` and `Page Down`, and `Up/Down Arrow` keystrokes can be used if the **Set** spin-button has focus. It is important to note that incrementing/decrementing the screen-set will *not* wrap around. We consider this a feature rather than a bug, at this time. There are some other important considerations for set-handling. See section 14 "Seq66 Set Master" on page 192.

Note the buttons for changing and showing the loop/recording modes of the grid slots and recording quantization. The normal arm/mute (loop) mode can be changed to other grid modes to make it easier to record, copy, paste, delete, and do other operations on a pattern.

Seq66's pattern grid can be put in various recording modes (e.g. overdub/merge versus overwrite) Quantization can be turned on globally in *Seq66*'s pattern grid as well. These operations are also available as automation controls. See section 12.5.4.28 "Automation / Record Modes and Quantization" on page 171.

5.1.1 Pattern Slots

A **pattern slot** is a box/button that can hold a pattern. If a pattern is present in the slot, text information and MIDI events are drawn on the button. The top line shows the title of the pattern, the number of measures in the pattern, and indicate if the pattern has a loop-count (indicated by a + sign. In addition, if the pattern has an input buss set, that number appears before the measures number; they're separated by a colon. A *right-click* over a pattern button brings up a fairly extensive popup menu.

When a pattern is left-clicked, in the default grid mode, its armed/muted status is toggled. This action can be changed via the "grid modes" discussed in section 5.1.3.5 "Pattern Loop-Record Modes" on page 70.

A pattern can show a number of different statuses based on the coloring of elements in the pattern slot. (However, some of the special coloring using in *Sequencer64* are not supported in *Seq66*).

- **Empty background**. When the default button coloring for the current *Qt* theme is shown, without a pattern box, this state indicates that the slot is unused.
- **Yellow pattern box**. This color is used when a pattern is first created by *double-clicking* on the slot. However, this color sticks even when notes are added. Feel free to change it to another color, or no color.

- **Normal background**. Unarmed (muted) patterns show the unactivated/unchecked state of the button as per the *Qt* theme. If a color is applied, it has a slight bit of alpha in the color so that the color appears muted.
- **Active background**. An armed (unmuted) pattern shows the activated/checked state of the button as per the *Qt* theme. If a color is applied, it has no transparency, and the color appears bright.
- **Line events**. Lines indicate the presence of notes. Depending on settings, the lines indicate the notes themselves, or a "fingerprint", a condensed indication of notes useful in reducing the overhead of drawing long patterns.
- **Red events**. Indicates a pattern for which the transpose feature is disabled. Most useful with drum patterns.
- **Circular events**. Small circles indicate tempo events. Generally, these events should appear only in the tempo track (which is normally track 0).

The user can also apply coloring to each sequence. This feature was adopted from *Kepler34* [8]. The color is more saturated when the pattern is unmuted.

5.1.1.1 Pattern Context Menus

Right-clicking on an **empty slot** brings up a menu to create a new loop or open an external live grid, as well as some other operations. This is the menu for an empty slot:

1. **New pattern**
2. **External live frame for set 0**

1. New. Creates a new loop or pattern. Clicking this menu entry fills in the empty box with an untitled pattern. Another way to create a new loop (without using the menu), is to hold the `Ctrl` key and click on the slot. A third to create a new loop is to *double-left-click* on an empty slot; this also brings up an external pattern editor (discussed later).

2. External live frame for set 0. This option brings up an external **Live** frame window, which is the same as the patterns panel, but can be used to show a different set in a multi-set project. Up to 32 external live frames can be shown. An external live grid can be activated by the **Activate** button, which sets the playing screen-set (and updates the main window to match). Another way to bring up an external live frame is to hold the `Shift` key and click on the slot.

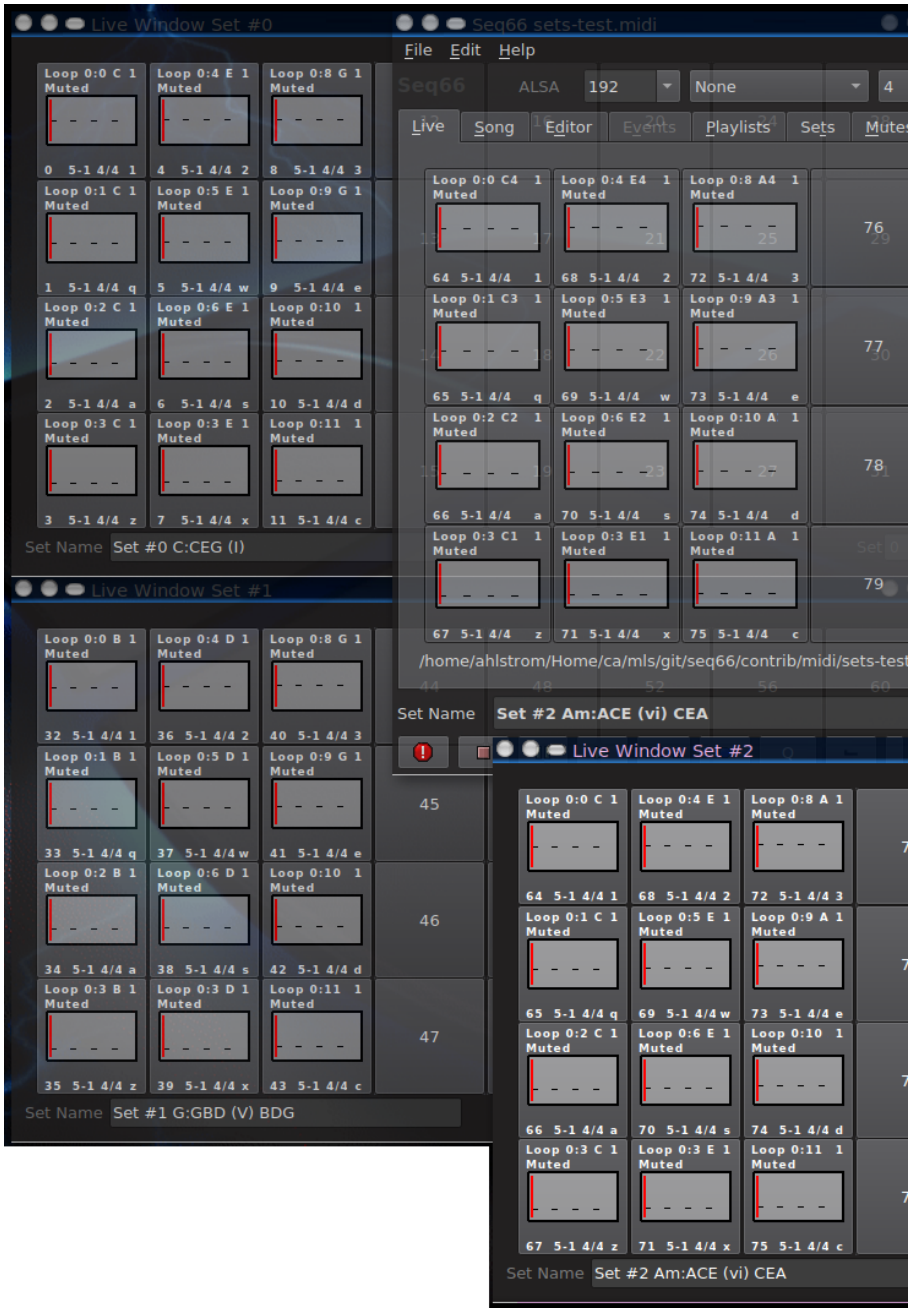


Figure 22: Multiple Live Grids

Note that the right-click slot menu has some items removed when the live grid is in an external window, or the main window is showing a set other than set 0, as indicated by the asterisks in the list below.

Once a pattern is created in a slot, there are more right-click options for that slot. A *right-click* on a **slot with a pattern** brings up a menu to allow one to edit it, or perform a few other actions specified in the context menu. Here are the menu entries:

The menu for a slot with a pattern is more extensive:

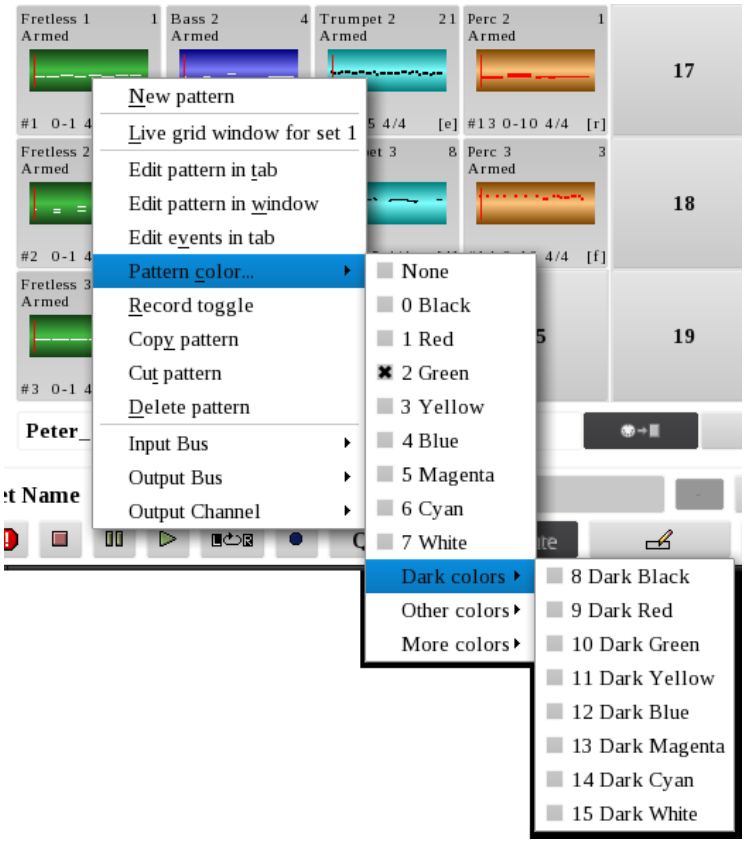


Figure 23: Patterns Panel Pop-up Menu

1. **New pattern**
2. **External live frame for set ...** *
3. **Edit pattern in tab** *
4. **Edit pattern in window** *
5. **Edit events in tab** *
6. **Pattern color**
7. **Record toggle**
8. **Copy pattern**
9. **Cut pattern**
10. **Delete pattern**
11. **Merge into pattern**
12. **Input Bus**
13. **Output Bus**
14. **Output Channel**

The first menu entry is the same as above. However, since there is already a pattern present in the slot, the user is prompted before erasing the current pattern and creating a new one.

1. New Pattern. Creates a new pattern in the empty slot. A double-click on an empty slot can also be used.

2. External Live Frame for Set. This selection uses the pattern number to open the

corresponding screenset number in an external **Live** frame. This allows viewing and interacting with a number of sets.

3. Edit Pattern In Tab. Selecting this item activates the **Edit** tab and fills it with data from the selected pattern. Note that this editor is slightly simpler than the external pattern editor described just below.

4. Edit Pattern In Window. Selecting this item brings up the pattern in an external pattern editor that has a few addition controls over the **Edit** tab (where space is more constrained).

In addition to *right-click* and select **New**, the user can *double-click* on the empty slot, to bring up a new instance of the sequence editor. For *double-click* on an existing pattern, the effect can be a bit confusing at first, because it also toggles the armed/muted status of the slot quickly twice (leaving it as it was).

A nice feature is hitting the equals ("=") key, then hitting a pattern shortcut key (hot-key), to bring up a new sequence or edit an existing one in a **Pattern Editor** .

5. Edit Events In Tab. Edits an existing loop or pattern, but using a detailed **Event Editor** tab that shows events as text and numbers, and allows editing them as text and numbers. This editor is basic, meant for viewing MIDI events and making some minor edits or deletes. The **Event Editor** is most useful when trying to find events that are screwing up the performance of that pattern. See section 8 "Event Editor" on page 108, for more information.

Another feature is hitting the minus ("-") key, then the hot-key, to bring up the **Event Editor** tab. The configuration file settings for the the '=' and '-' keys can be altered in the 'ctrl' file.

6. Pattern Color. Opens a menu to select a color for the pattern. This selects a color palette value (index) into the currently loaded color palette. 32 palette colors are supported, and the palette can be modified. See section 16 "Palettes for Coloring" on page 199.

7. Record Toggle. Selecting this item toggles the recording status of the pattern. If the grid record mode button shows something other than **None** (e.g. **Quantize**), then that alteration is supplied. Note that recording can also be set in the pattern editor, or via the "record" loop-mode, or by automation. (The default key is `+`; click it and then select the desired slot via mouse or hot-key.)

8. Copy Pattern. Copies the pattern underneath the mouse cursor. The pattern can then be pasted elsewhere in the Patterns panel. One can also drag-and-drop a pattern into another cell (there is no outline box during the drag, unfortunately). See section 5.1.1 "Pattern Slots" on page 58. Note that there is no `Ctrl-C` key for this operation in the live (main) window.

9. Cut Pattern. Cuts the pattern while copying it for later pasting. There is no `Ctrl-X` key for this operation.

10. Delete Pattern. Deletes the pattern. Currently the same as Cut!

11. Paste Pattern. Pastes a loop or pattern that was previously copied. This option is shown only when *right-clicking* over an empty pattern. It causes a cut or copied pattern to be replicated into the emptly slot. Note that there is no `Ctrl-V` key for this operation in the main window. Also note that the pattern clipboard can be pasted after a **File** / **New** or **File** / **Open**, to copy it to another file.

12. Merge Into Pattern. Like **Paste to pattern**, it pastes a patten that was cut or copied into the pattern slot where the mouse was *right-clicked*. However, the original notes remain. Thus, the merge option provides a way to build up a pattern by copying other patterns.

13. Output Bus. This item allows one to select the output buss for a pattern without having to open the editor. Note that having an output buss setting for a pattern is mandatory. The output buss is stored as a simple buss number ranging from 0 on up to the number of MIDI output devices.

14. Output Channel. This item allows one to select the output channel for a pattern without having to open the editor.

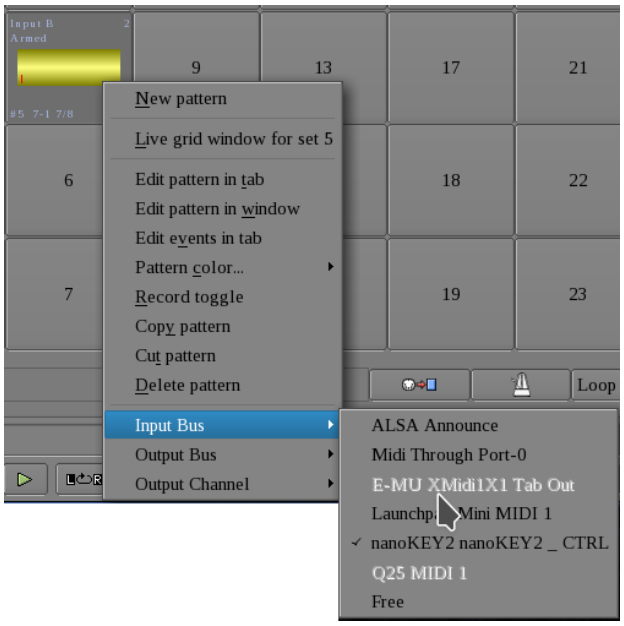


Figure 24: Input Bus Popup Menu

15. Input Bus. *Seq66* supports an *optional* input buss setting for each pattern. This setting is not available in the pattern editor due to lack of room, so this popup menu is the only way to get to this option. Normally, the "current" pattern, which is the one opened in the pattern editor, is the only one accepting input when recording is activated.

By setting an input buss, incoming MIDI events can be routed to the pattern that is set to accept input from that buss. As the figure shows, the full set of input ports is shown. Some are disabled. (The "ALSA Announce" buss, should be disabled; this is a bug.)

In the figure, the "Input B" pattern will accept events from the *KORG nanoKEY2* keyboard. If the **Free** entry is selected (the default), then the pattern will accept input from any MIDI input port.

Important:

- If a MIDI input device is not enabled in the port list, it will not work as an input buss.
- If a MIDI input device is defined as an automation controller for *Seq66*, it will not work as an input buss.
- To disable the usage of input-buss routing, set the input buss to **Free** for all patterns, then

save and reload the file.
- To record normally to a pattern, open the pattern for recording and use a MIDI device that has *not* been assigned as a pattern's input buss.

One final note... the input buss setting overrides the record-by-channel setting.

5.1.1.2 Pattern Labels

A **filled pattern slot** is referred to as a *pattern* (or *track*, *loop*, or *sequence*). A pattern is shown in the pattern grid as a filled slot with a number of labels surrounding it. Here are the labels shown:

- **Pattern Name**. Top left. This line, in the upper left of the pattern slot, contains the name or title of the pattern, for reference when juggling a number of patterns.
- **Pattern Status**. Second line. Underneath the pattern-name is the status of the pattern, such as "New" (untitled with zero events), "Armed", "Muted", or "Queued". This status is useful when the Qt theme coloring makes the exact status difficult to determine.
- **Pattern Length**. Top right. The length of the pattern, in measures, is shown in the upper right corner of the pattern slot. If the pattern has been modified, an **asterisk** appears after the number. Otherwise, if the loop-count for the pattern is greater than 0, then an **plus sign** is shown. Remember that a pattern loop-count of 0 means the pattern can repeat "forever".
- **Notes or Fingerprint**. Center progress box. The contents of the pattern, in the central box, provide a distinguishable representation of the notes or events in the pattern. The notes are shown in the center, inside a "progress box" that can also be colored, or not shown at all. Long patterns can be replaced by a much shorter "fingerprint", for faster drawing. Tempo events are indicated by a small circle. An pattern with no playable events will not needlessly scroll during playback. However, if a pattern has even a single event (say, a program change), it will scroll.
- **Progress Cursor**. Center. At the left of each center box is a vertical line, waiting for playback to start so that it can move through the pattern, again and again. When the song is playing, this vertical bar tracks the position of the playback of the pattern or loop; it returns to the beginning of the box every time the pattern starts over.
- **Sequence Number**. Bottom left. This number is shown at the bottom left of the pattern slot. Pattern numbers, by default, range from 0 to 1023. Note how it varies fastest by row (top to bottom). There is a configuration item to change transpose them.
- **Bus-Channel**. Bottom, second from left. This pair of numbers shows the the MIDI buss number, a dash, and the MIDI output channel number. For example, "0-2" means MIDI buss 0 (re 0), channel 2 (re 1). If the channel is an "F", this means that the pattern has no specified output channel, and can play on all channels. This "free" channel concept is useful for applying Program Changes and Volume controls to many channels at once.
- **Beat/Beat Width**. Bottom, third from left. This pair of numbers is the standard time-signature of the pattern, such as "4/4" or "3/4". The first number is the beats-per-measure, and the second is the size of the beat, here, a quarter note.
- **Shortcut Key**. Also known as the **hot key**, this symbol is shown at the bottom right of the slot, in square brackets for better visibility. The key noted in the lower-right corner of the pattern is a "hot-key" that can be pressed to toggle the mute/unmute status of that pattern. This action is an alternative to *left-click* on the pattern. This hot-key can also be used to

open the pattern in a pattern editor or in the event editor. Other actions are supported by changing the **loop mode**.

- **Armed**. Highlight color of button. Button highlighting indicates that the pattern is armed (unmuted), and will play if playback is initiated in the pattern window in live mode. An item is armed/disarmed by a *left-click* on it, or by using the button's hot-key.
- **External Frame**. *Shift-left-click*. If the `Shift` key is held during a *left-click* on a pattern, the corresponding set's **Live** frame is brought up.

Left-click on an filled pattern box will toggle the status of the pattern between muted (white background) and unmuted (black background). If the song is playing via the main window, toggling this status makes the pattern stop playing or start playing. The armed status can also be toggled using hot-keys and MIDI controls.

5.1.2 Basic Pattern Control

In addition to the arm/mute function done by clicking on a pattern slot, there are some additional controls that can be enabled on the fly:

- **Queue**. Enables one pattern to be turned on or off as soon as the pattern loops to its beginning.
- **Keep-queue**. The same as Queue, but stays active until disabled.
- **Oneshot**. Similar to Queue, but it merely unmutes a pattern for one cycle, then mutes it.
- **Replace**. Soloes a pattern immediately. Can be repeated many times. Then restores the original status of patterns.
- **Solo**. Similar to Replace, but the process is queued.
- **Snapshot**. Similar to Queue, but arms a pattern, plays it once, and turns it off.
- **Learn**. Maps a set of pattern arm/mute statuses to a keystroke.

These actions are described in detail below. They have some common preconditions and indicators that are listed here:

- **Live mode**. Make sure that the song is in **Live** mode.
- **Playback**. These features work best with playback already started.
- **Enabled patterns**. Make sure that some patterns are armed. In some cases, a snapshot of them is saved.
- **Status indicator**. When a control-mode is activated, a status indicator appears, with italics text indicating the status. It appears between the file-name and the Metronome button.

The keystrokes noted are mentioned in section 5.1.3.1 "Pattern Keys" on page 68.

5.1.2.1 Pattern Queue

Pressing the "queue" key and then hitting a pattern hot-key will queue an on/off toggle for a pattern when the end of the loop is reached. This means that the change in state of the pattern will not take hold immediately, but will kick in when the pattern loops back to its beginning. This pending

state is indicated by coloring the central box of the pattern grey. Also check out the "keep queue" functionality and the "one-shot queue" functionality described below.

1. Press `o`; note that a "Queue" status indicator appears.
2. Click a pattern.
3. It is queued for arm or queued for mute (the opposite of its original status).
4. It is in effect for only one pattern click, then must be reestablished, if desired, by going back to step 1.

5.1.2.2 Pattern Keep-Queue

Pressing the "keep queue" hot-key activates a "sticky" queue mode. In this mode, pressing a pattern key immediately turns on queuing. Multiple patterns can be handled serially in this way; keep-queue persists until one clicks the queue hot-key again, or changes the active (viewed) screen-set. Keep-queue" is cancelled by pressing the keep-queue hot-key again. There is also a **Q** button in the main window for the same purpose. Also note the "queued replace/solo" functionality, described a bit later.

1. Press the backslash; note that a "Keep Q" status indicator appears.
2. Click a pattern.
3. It is queued for arm or queued for mute (opposite of original status)
4. It is in effect for all pattern clicks until the keep-queue key is pressed again. The "Keep Q" status indicator disappears.

5.1.2.3 Pattern Oneshot

Thanks to *Kepler34*, we have "one-shot queue" functionality. This one-shot setup queues a pattern up for unmuting only, and, once the pattern has played, it is automatically muted. This process is easier than having to unqueue the pattern manually before the next playback.

A light-grey color is used to show a "one-shot" queue. The one-shot queue can only turn a pattern on, and it will force the pattern off after one play.

1. Press the pipe key; note that a "Oneshot" status indicator appears.
2. Click a pattern.
3. It is queued for arm only.
4. It plays once, and then stops. The "Oneshot" status indicator disappears automatically.

5.1.2.4 Pattern Replace

The "replace" automation key/control sets a form of muting/unmuting. When the "replace" hot-key is pressed, and then a pattern's hot-key is pressed, that sequence is unmuted, and all of the other sequences are muted. Other patterns can be clicked to be armed/muted; when the "replace" hot-key is clicked again, the original patterns armed before the first "replace" click are restored. "Replace", like other controls, is also implemented via MIDI control, where the MIDI control can be activated; but then the user has to select the desired sequence.

1. Press 0; note that a "Replace" status indicator appears.

2. Click a pattern; it either stays on or is turned on.
3. All other patterns are saved as a snapshot, then are immediately muted (no queuing).
4. More patterns can be soloed by clicks.
5. Press 0 again, and the original patterns (snapshot) are restored. The "Replace" status indicator disappears.

5.1.2.5 Pattern Solo

Seq66 provides an extension to the replace functionality that is called "queued-replace" or "queued-solo". In this feature, the "keep queue" function is activated and the replace function is queued so that it does not occur until the next time the patterns loop. And queued-replace provides a form of snapshot, limited to the *current* screen-set. Here are the steps:

1. Be careful which key, "BS" (`Ctrl-H`) or "BkSpace" (`Backspace`), is configured. They are two different keystrokes. "BkSpace" is to be preferred and is the default.
2. Start playback with some patterns on.
3. Press and release the "solo" hot-key. This puts the application into "queued replace" mode. It is indicated via a highlighted "**Q**" button and a "Solo" status indicator.
4. Click the desired pattern hot-key. Observe that it arms or stays on, and that the other playing patterns show the "queued" color (grey). At the end of the loop, they turn off, and the "replace" pattern is now soloed. This step can be repeated with other patterns.
5. To end the "queued-replace" mode, click the "solo" hot-key. This restores the original statuses of the patterns. Also, changing the active screen-set ends "queue-replace" mode. to solo.

5.1.2.6 Pattern Snapshot

When a snapshot key is pressed and released the state of the patterns (armed versus unarmed) is saved as a snapshot. While the snapshot is in force, one can then change the state of the patterns (using keyboard, mouse, or MIDI controls) to change how the song plays. When the snapshot mode is exited by pressing the snapshot key again, the original saved state of the patterns is restored.

1. Press `Ins`; note that a "Snapshot" status indicator appears.
2. The first press of Ins saves the current state of the patterns.
3. The user can mute/unmute any combinations of patterns.
4. The next press of `Ins` restores the snapshot. The "Snapshot" status indicator disappears.
5. The snapshot "buffer" is cleared to get ready for the next `Ins`.
6. Snapshot mode is non-queued.

5.1.2.7 Pattern Learn

Learn is activated by the `l` (lower-case L) key or the "L" button. Once activated, the next pattern hot-key (lower-case) pressed will cause the current set of armed/muted patterns in the screen-set to be stored with the upper-case (shifted) version of that hot-key. See section 3.5.9 "Mute Group Learn" on page 27.

5.1.3 Pattern Keys and Clicks

This section recapitulates all the clicks and keys that perform actions in the Pattern windows. Some additional clicks and keys are noted here as well. Also refer to section 17 "Seq66 Keyboard and Mouse Actions" on page 202.

5.1.3.1 Pattern Keys

Each pattern in the patterns panel can have a hot-key or shortcut-key associated with it.

Pattern Toggle. Like a *left-click*, for each pattern, its assigned hot-key will also toggle its status between muted/unmuted (armed/unarmed). Here is the normal layout of patterns, which was built into *Seq24*'s "DNA":

```
[  0 ] [  4 ] [  8 ] [ 12 ] [ 16 ] [ 20 ] [ 24 ] [ 28 ]
[  1 ] [  5 ] [  9 ] [ 13 ] [ 17 ] [ 21 ] [ 25 ] [ 29 ]
[  2 ] [  6 ] [ 10 ] [ 14 ] [ 18 ] [ 22 ] [ 26 ] [ 30 ]
[  3 ] [  7 ] [ 11 ] [ 15 ] [ 19 ] [ 23 ] [ 27 ] [ 31 ]
```

There is an alternate (and fairly new) mapping that can be enabled using the `swap-coordinates` option in the 'usr' file. See section 12.4.3 "'usr' File / User Interface Settings" on page 158. Below is the default keyboard "grid" that is mapped to the loops/patterns on the screen-set.

```
[ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ][ 8 ]
[ q ][ w ][ e ][ r ][ t ][ y ][ u ][ i ]
[ a ][ s ][ d ][ f ][ g ][ h ][ j ][ k ]
[ z ][ x ][ c ][ v ][ b ][ n ][ m ][ , ]
```

These characters are shown in the lower right corner of each pattern, as an aid to memory. This grid can be changed in the 'ctrl' file in the `[loop-control]` section. However, it is best to leave this setup as is, except for the key-swapping needed for alternative keyboard layouts like "QWERTZ".

Pattern Shift. A "shift" functionality is available for the mute/unmute hot-keys when a set is larger than 32 patterns. Normally, pressing the `1` key will toggle sequence 0. If preceded by one slash key (`/`), then sequence 32 will be toggled. If preceded by two slash keys, then sequence 64 will be toggled. This features supports using set sizes of 32, 64, and 96 patterns.

Screenset Increment and Decrement. The `"["` and `"]"` keys on the keyboard decrement or increment the set number.

Snapshot. Unlike in *Seq24*, the `Alt` keys are not used for snapshots. In addition, the snapshot key acts like a *toggle*... no need to hold it down. It's best to use something that does not trigger desktop/window commands; the default key is `Ins`. Configured in the 'ctrl' file `[automation-control]` section.

We do **not** recommend using `Ctrl` or `Alt` keys for pattern control. They conflict with application or desktop settings. The default queue key is `o` (lower-case O).

The default keep-queue key is the `Backslash`.

The default one-shot queue key is `|`, the pipe character.

The default keep-queue key is `Keypad Home`.

The default solo key is `Backspace`.

There are more keys defined in the 'ctrl' file, and it is worth figuring out what they do, if not documented here. Also see the installed `control_keys.ods` spreadsheet. For a couple of short, but good, video tutorials about using arming, queuing, and snapshots in *Seq24*, see reference [42].

5.1.3.2 Other Pattern Clicks

Left-click on a pattern-filled box will change its state from muted (white background) to playing (black background), whether the sequencer is playing or not.

Left-click-hold-drag on a pattern, drags it to a different pattern on the grid. The box disappears while dragged, and reappears in the new location when dropped. However, a pattern *cannot* be dragged if its **Pattern Editor** window is open.

Right-click on a pattern brings up the appropriate context menus, as discussed earlier, depending on whether the pattern box is empty or filled.

Middle-click on a pattern will bring up the pattern in the **Editor** tab in the main window.

Shift-left-click on a pattern will open up an external live-frame for the set having the same number as the pattern.

Ctrl-left-click on a pattern will create a new pattern, just like double-click will (if enabled).

There is a truer "Solo" functionality in the Patterns Panel and the Song Editor. To "solo" a pattern, move the mouse cursor over the pattern, hold the `Alt` key, and left-click the pattern. This will turn off all the other patterns, so that the selected pattern ins the only one playing. Also see section 5.1.3.5 "Pattern Loop-Record Modes" on page 70; it notes another way to solo a pattern.

One thing to remember with solo is that it can be reversed (reenabling all the other patterns) only if that same pattern is clicked. Soloing another pattern will lose the list of previously-enabled patterns.

5.1.3.3 Metronome

Also shown in the figure is a **Metronome** button. This is a feature new with *Seq66* version 0.99.0, and still has a few minor issues to work out, but it works. The metronome has a number of configurable items. See section 12.3.16 "'rc' File / Metronome" on page 151.

Clicking the metronome button turns the metronome on and off. To use the metronome (once configured), click the button to enable it, then start playback. It can be turned on and off during playback. It is always available, stored as a hidden automatically-generated pattern.

5.1.3.4 Count-in Recording

Also shown in the figure is a **Count-In Recording** button. This feature still has a few minor issues to work out, but it works. It is not **background recording**, which involves a thread continuously recording in the background. Some day.

The recorder has a number of configurable items. See section 12.3.17 "'rc' File / Count-In Recording" on page 151. Count-in recording needs to be enabled in the configuration. Otherwise, the count-in-recording button is disabled.

When enabled, press the count-in-recording button. Now all events on the given input buss will be recorded, even if playback is not started. Better to start playback after enabling recording!

Once one is satisfied with the recording, stop playback, then click the count-in-recording button again. If any events have been recorded, then they are pasted into a new pattern in the next available pattern slot. There, they can be edited or deleted. Note that the count-in pattern is muted while recording. We will change how it works based on user feedback.

5.1.3.5 Pattern Loop-Record Modes

Seq66 has always had the feature of turning on recording, including quantized recording, in the pattern editor. Now, it also provides for loop-grid modes. With this feature, the normal loop muting/unmuting functionality of the **Live Grid** buttons can be changed to allow the toggling of recording or other actions to be applied to the affected pattern. This new functionality is enabled by two buttons in the main window grid and by two refactored keystroke/MIDI controls ("Record" and "Quan Record"). The normal mode (mute/unmute) is called **Loop**.

- **Loop**. This mode is the legacy and long-time standard mode of *Seq24*, *Sequencer64*, and *Seq66*. When in this mode, a click on a pattern slot, a loop-control keystroke, or a loop-control MIDI event will change the mute/unmute, armed/unarmed status of a pattern.
- **Record**. A click/key on the pattern turns on recording for that pattern. This status is shown by the appearance of a red circle in the pattern button.
- **Copy**. Copies the selected pattern into the clipboard.
- **Paste**. Pastes the clipboard into the selected pattern.
- **Clear**. Removes the events from the selected pattern. Careful!
- **Delete**. Deletes the selected pattern. Careful!
- **Thru**. Turns on the MIDI Thru functionality of the selected pattern.
- **Solo**. Soloes the selected pattern.
- **Cut**. Deletes the selected pattern and copies it into the clipboard.
- **Double**. Doubles the length of the selected pattern.

It has been supplemented by four recording modes. Here are the modes:

- **Overdub**. Also known as **Merge**, this mode is selected by clicking on the **Loop** button, then using the **Record** control. These commands cycle between all the modes in this list. Overdub is the same as the **Merge** option in the pattern editor. Clicking a grid will turn on/off the overdub recording mode for that pattern. It enables recording that accumulates note events in each pass of looping through the pattern.

- **Overwrite.** This mode causes the grid button to turn on overwrite recording. When the loop restarts over and a note is pressed, then the existing notes in that loop are erased, and the new note is added.
- **Expand**. This mode causes the grid button to turn on expand recording. Once the end of the loop is near, whether or not any notes are being input, another measure is added to the length of the loop.
- **One-shot**. When this option is set, with the record button on, and no pattern playing, recording won't start until a note comes in, and when the first note comes in, the progress bar starts at the left (time 0). As each new set of notes at the same timestamp come in, the notes are recorded and the current time advances by one snap value. Once the end of the pattern lenght is reached, recording is turned off.
- **One-shot Reset**. When the **One-shot** option is selected, this new option is enabled. When clicked, three things happen:
  - All the (note) events just recorded are *erased*.
  - One-short counters/flags are *reset*.
  - Recording is turned *back on*. Thus, one can easily re-do a one-shot recording.

These loop-record modes are automatically applied if a pattern is opened in the pattern editor.

Once the recording mode is turned on, whether here or in the pattern editor, then the recording type comes into play. The types of alteration that can come into play are set by clicking the button at the right labeled "None" until the desired recording mode is shown:

- **None**. Incoming events are recorded as is. Plain recording, events recorded with timestamps unaltered. Indicated by a red circle in the pattern slot.
- **Quantize**. When recording, quantize the incoming events. Incoming events are quantized to the snap value of the pattern. Indicated by a red circle and a "Q" inside it in a pattern slot.
- **Tighten**. A weaker version of **Quantize**. Incoming events are partially quantized to the snap value of the pattern. Indicated by a red circle and a "T" inside it in a pattern slot. This mode is accessible only via this recording mode; there is no such button in the pattern editor.
- **Note Map**. If a 'drums' file has been specified and is active, then incoming events are remapped to different notes. This feature is useful in transforming the notes of an old drum machine into, say, a *General MIDI* drum kit. Indicated by a red circle and an "N" inside it in a pattern slot. This mode is accessible only via this recording mode; there is no such button in the pattern editor.

These alterations are automatically applied if a pattern is opened in the pattern editor.

5.2 Patterns / Bottom Panel

The first line of the bottom panel contains the **Set Name** of the current set, which is editable. It contains a button with an exclamation point (**!**), which can be used to reset the play-set (the total of all patterns that are loaded for playback) to the patterns in the current set. The **Set** spin-box can be used to change the current set.

1. **Set Name**

2. **Set Reset**
3. **Set**

1. Set Name. Each of the 32 available screen-sets can be given a name by entering it into this field. This name is saved with the MIDI file.

2. Set. This spin widget selects the current screen-set. The values in this field range from 0 to 31 (less if the set-size is a larger value), and default to 0.

3. Set Reset. *Seq66* has a new feature whereby multiple sets can play at once. This button, an exclamation point, will reset the playback to the patterns in the current set.

The bottom panel of the Patterns window provides way to control the overall playback of the song. It has changed quite a bit over the last few versions of *Seq66*, and we have not yet caught up with the diagrams. And the Qt user-interface adds more changes. Refer to the diagram of the whole window, for now. It has a number of items:

1. **Panic!**
2. **Stop**
3. **Play and Pause**
4. **Loop**
5. **Song Record**
6. **Keep-Queue Status**
7. **Toggle Song/Live Mode**
8. **Tap Tempo**
9. **BPM**

1. Panic!. This new button stops the song and sends MIDI Off messages on all notes.

2. Stop. The red square button stops the playback of the song and all its patterns. The keystroke for stopping playback is the `Escape` character; it stops playback and rewinds to the beginning of the song. The `Space` keystroke will do the same thing if playback is in progress; it is effectively a playback toggle key.

3. Play and Pause. The green triangular button starts the playback of the whole song. The keystroke for starting playback is the `Space` character by default. It also stops playback, also rewinding the song to the beginning. The Pause button toggles playback without rewinding the song. A Pause key (by default, the period) is also defined.

4. Loop. If this button is active, then the playback will loop between the "L" and "R" markers in the pattern editor or the song editor time-lines. As with the "L" and "R" markers in the pattern editor, this can be placed via left and right mouse clicks. Note that the "L" and "R" markers can be selected via the keyboard using their respective shifted key. Once selected, the marker can be moved left or right using the left and right arrow keys.

5. Song Record. Song-recording in *Seq66* is adopted from the *Kepler34* project. This feature takes live muting changes and records them as triggers in the **Song Editor**. The default hot-key for this function is `P`. This feature does not honor queuing... rather than waiting until the end of the pattern when the queuing takes effect, the trigger recording starts immediately.

6. BPM. The spin widget adjusts the "beats per minute" (BPM) value. The range of this field

is from 1 bpm to 600 bpm, with a default value of 120 bpm. Although this field looks editable, it is not. Most keystrokes that are entered actually toggle one of the pattern boxes. However, the following keys can also modify the BPM in small increments: The `semicolon` reduces the BPM; The `apostrophe` increases the BPM. Also, if one *right-clicks* on the **Up** button, the BPM advances to its largest supported value, and if one *right-clicks* on the **Down** button, the BPM advances to its lowest value. MIDI control for this value is also available. The precision of the BPM value can be set to 0, 1, or 2 decimal places, and the increment values for the step size (small) or page size (large) of the BPM spinner can be configured in the 'usr' file.

7. Tap Tempo. This control is clicked in time with a tune, to set the tempo based on the tempo of the clicks. Once clicked, the label of this button increments with every click, and the **BPM** field updates to display the calculated tempo. If the user stops tapping for 5 seconds, the label reverts to 0, the BPM value keeps its final value, and the user can try tapping the tempo again, or accept the current value. Tapping can also be done using the keystroke defined in the 'ctrl' file. It defaults to the "`F9`" key.

8. Keep-Queue Status. This item is the **Q** button. It provides a visual way to know the current state of keep-queue, and is activated either by clicking on it or by pressing the assigned keep-queue key.

9. Toggle Song/Live Mode. Pressing this button toggles between the song mode and live mode. Note that the default mode is configurable, and *Seq66* can go into song mode when a loaded MIDI file has triggers, if configured to do so.

5.3 Patterns / Multiple Panels

Multiple patterns-panels can be created in addition to the one in the "Live" tab. The live set-up and set-down keystrokes, as well as their MIDI control counterparts (both defined in a 'ctrl' file). apply only to the main window.

5.4 Patterns / Variable Set Size

This option, informally known as "variset", allow some changes in the set size and layout from the default 4x8 = 32 sets layout. The row count can be set from 4 to 8, and the column count can be set to 8 to 12. Note that the set size can only be *increased* by these settings.

Warning: *seq24* was fairly hardwired for supporting 32 patterns per set, and there are still places where that is true. Thus, consider this option to be experimental.

The `-o sets=8x8` option can be used to set this mode. These settings can be made permanent in the 'usr' file. In that file, the options modified are `mainwnd_rows` and `mainwnd_cols`.

Generally, it is recommend to stick with the 4x8 (32 patterns/set), 8x8 (64 patterns/set), and 8x12 (96 patterns/set). This works best with the existing set of 32 hot-keys.

Also note that the Qt 5 user-interface also supports "variset", whether in the main window or in the external live-frame. In addition, the Qt windows can be resized and still show reasonable renditions of the pattern-slots.

5.5 Patterns / Set Handling

Let's go through an example using the `Home` key (or whatever key is configured as the **Set Playing Screenset** key.)

1. Load a song with more than one screen-set.
2. Unmute the pattern(s) in the first set and start playback.
3. Use the "]" (**Screenset Up**) key to move to the next set. Note that the first set is still playing. Also note that the now-current set is *not* playing.
4. Press the `Home` key. Note that the first set turns off, and the current set turns on. These steps can be repeated at will.
5. Finally, hit the `F8` (**Toggle Mutes**) key. Note that all tracks on all sets toggle muting each time this key is pressed.

6 Pattern Editor

The *Seq66* **Pattern Editor** can edit and preview a pattern, configure its buss, channel, transpose, musical scale, and many other settings. A slightly modified version of the **Pattern Editor** appears in the **Edit** tab in the main window; the main window has to be expanded vertically to see all of the controls. The complete version can be brought up in an external window.

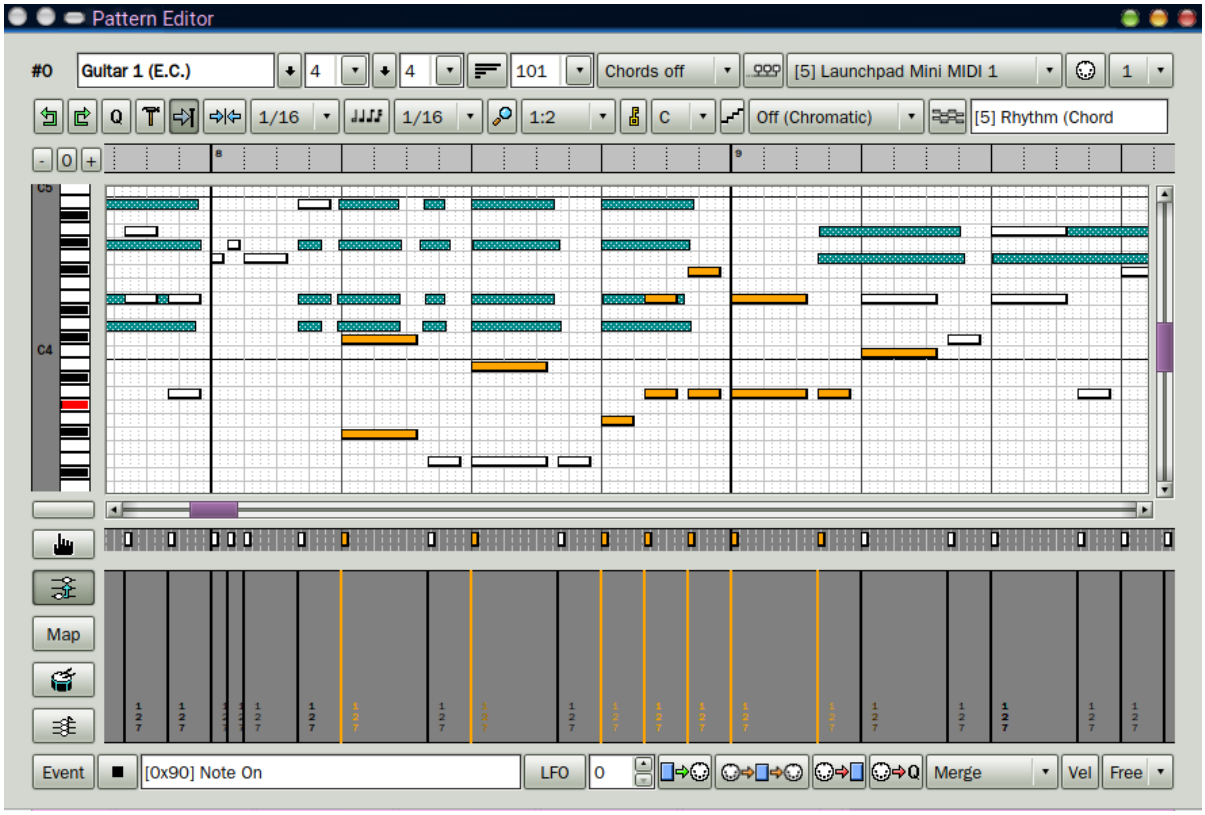


Figure 25: External Pattern Editor Window

This figure does not show a recent new feature: for selected event, the orange data line is topped with a circular "grab handle" that can be used to change the amplitude of the data (e.g. velocity) being shown.

The **Pattern Editor** is complex, and we will discuss the external window only since its features are a superset of the **Edit** tab. For exposition, we break the window into the following sections:

1. **First Row**
2. **Second Row**
3. **Time**
4. **Piano Roll**
5. **Events Pane**
6. **Left Buttons**
7. **Data Pane**
8. **Bottom Row**
9. **Common Actions**

Before we describe this window, there are some things to recognize. First, if the pattern is empty when play is started, the progress bar will still move, so that the user can play a MIDI instrument and record new notes. Second, to add a note with the mouse, one must press the *right* mouse button (the pointer changes to a pencil) and, *while holding it*, press the left mouse button. Or click in the pattern editor, press the `p` key to select the "pencil" or "paint" mode, or `i` key to select "insert" mode (a la vi), then left-click to add a note or left-click-drag to add multiple notes as the mouse moves. Press or release the right mouse button, or press `x` to "eXit" or "eXscape" from paint mode. `Esc` also exits paint mode if the pattern is not playing. Another option is to press the "finger" button to toggle between note-entry and note-selection. Third, notes are drawn only with the length selected by the "notes" button near the top of the pattern window. There are tricks to modifying the new notes that are described later.

Seq66 automatically scrolls horizontally through the sequence/pattern editor window when playback moves the progress bar outside of the current frame of data. This feature makes it easier to follow patterns that are longer than a measure or two. One might want to print out the following figure to follow along. There is a lot of functionality in this window.

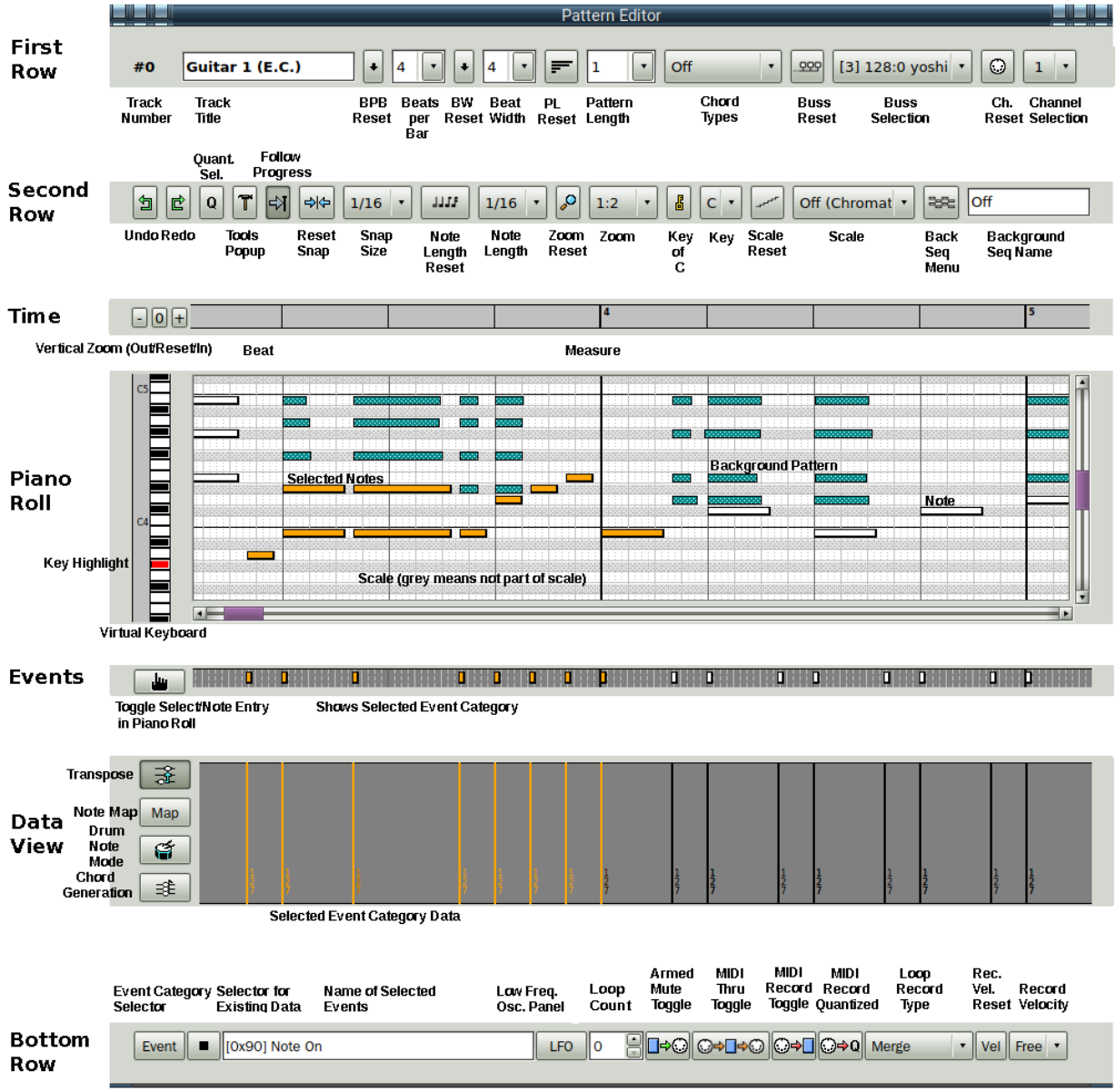


Figure 26: Pattern Editor Window, Annotated

The arrow keys can be used to move left, right, up, or down.

For *vi* users, the "h" (left), "j" (down), "k" (up), and "l" (right) keys can be used while any pane is in focus, and they act like the arrow keys.

6.1 Pattern Editor / First Row

The top bar (horizontal panel) of the Pattern (sequence) Editor lets one change the name of the pattern/loop/sequence/track, the time signature of the piece, how long the track is, and some other configuration items.

1. **Track Number**
2. **Track Name**
3. **Add Time Signature Event** (see composite picture below)
4. **Beats Per Bar Reset** and **Beats Per Bar**
5. **Beat Width Reset** and **Beat Width**
6. **Pattern Length Reset** and **Pattern Length**
7. **Chord Types**
8. **Buss Reset** and **Buss Selection**
9. **Channel Reset** and **Channel Selection**

1. Track Number. This item shows the sequence/track/pattern/loop number, to make it easier to pick it out when a lot of patterns are being edited at once.

2. Track Name. Provides the name of the pattern. This name should be short and memorable. It is displayed in the **Live Grid** (the **Patterns Panel**), on the top line of its pattern slot.

3. Add Time Signature Event. This button is shown in the composite picture in section 6.3 "Pattern Editor / Measures Ruler" on page 83, to the left of the beat selector. When either the beats/bar or beat width values (see below) are changed, and the progress marker is at time 0, then the change is saved as a Time Signature Meta event at time 0. Additional changes there overwrite that event. There are interactions between the beats/bar, beat-width, and the length (in measures) of the pattern. Generally, if both the beats/bar and beat-width are to be changed, it is better to change the beat-width first. In any case, the measures might have to be adjusted as well. It is better to get all the time signatures in place before adding notes. To add time signatures at later times in the tune, this must be done:

1. Move the mouse cursor to the *top half* of the time pane; the cursor will change to a vertical arrow. Click at the desired time and observe the red progress line.
2. Change the beats-per-measure, if desired.
3. Change the beats-width, if desired.
4. Clock the **Add Time Signature** button to log the new time signature.

It adds a time-signature Meta event and adjusts the pattern length (if needed) at once. The new time signature should appear in the data and event panes as well.

4. Beats Per Bar. Specifies the number of beat units per bar in the time signature. The possible values range from 1 to 16, if the drop-down menu is used. Arbitrary values up to 32 can be entered by typing the number. The "Reset" button resets the value to 4.

5. Beat Width. Specifies the size of the bottom beat unit of the time signature: 1 for whole notes; 2 for half notes; 4 for quarter notes; 8 for eight notes; 16 for sixteenth notes; and 32 for thirty-second notes. The whole time signature is display at the bottom center of the corresponding pattern slot in the **Live Grid**. Arbitrary values up to 32 can be entered by typing the number. The "Reset" button resets the value to 4.

Now, although the combo-box provides only beat-widths that are the MIDI-standard power of 2, the number can be edited for any number. A warning prompt appears, and if OK'ed, then the non-standard beat-width is set, but it is not added as a time-signature event. Instead, it is stored as a SeqSpec event for *Seq66*.

6. Pattern Length. Sets the length of the current pattern, in measures. The possible values

range from 1 to 64. Arbitrary values up to 1024 can be entered by typing the number. *However*, when opening or importing a non-*Seq66* MIDI tune, the length of each track will be used, and so other values are possible.

Bringing up a pattern less than one measure or bar in length in the pattern editor will adjust the pattern to pad it to the length of one measure. *Seq66* will, when it reads such a short pattern from a MIDI file,

A feature from user *stazed* allows the pattern to expand indefinitely while the user inputs MIDI from a controller, via the **Expand** option of the **Loop Record Type**. This works in **Live** mode or **Song** mode. The pattern is not extended until events come in. Thus, the pattern is only as long as the last measure when a note comes in.

7. Chord Types. This setting allows one to select a chord type (e.g. "major" or "minor"). When active, a note is treated like the base note of the selected chord type, and extra notes are generated to create that chord. The **Chord Generation Reset** button is at the left of the **Data Pane**.

8. MIDI Out Device (Buss). This setting specifies a virtual MIDI output buss or a MIDI output device set up by the computer and attached MIDI equipment. The button resets it to buss 0. Note that, if the pattern's selected buss is not found, this entry will be blank. The user must select a valid buss from this dropdown.

9. MIDI Out Channel. The **Channel Selection** setting selects the MIDI output channel. The possible values range from 1 to 16, plus *Free*, which means that the channels of the events are preserved, and are used as the output channel, a bit like an SMF 0 track. The editing channel is always channel 1. If instruments are assigned in the 'usr' configuration file to that device and channel, their names will be shown in the dropdown.

In addition, this setting determines the channel applied when painting notes in the piano roll. If set to "1" or "Free" (no channel), then channel 1 is applied. Otherwise, if set to "2" through "16", that channel is applied.

6.2 Pattern Editor / Second Row

The second horizontal panel of the Pattern Editor provides a number of additional settings and functions:

1. **Undo**
2. **Redo**
3. **Quantize Selection**
4. **Tools Popup**
5. **Follow Progress**
6. **Reset Snap** and **Grid Snap**
7. **Note Length Reset** and **Note Length**
8. **Zoom Reset** and **Zoom**
9. **Key Reset** and **Key of Sequence**
10. **Scale Reset** and **Musical Scale**
11. **Background Sequence**

1. Undo. The **Undo** button rolls back any changes to the pattern from this session. It will roll back one change each time pressed. Pressing `Ctrl-Z` is the same as using the **Undo** button.

2. Redo. The **Redo** button will restore any undone changes to the pattern from this session. It will restore one change each time it is pressed. There is currently no redo key.

3. Quantize Selection. This button quantizes the selected events as per the **Grid Snap** setting.

4. Tools. This button brings up a nested menu of tools for modifying selected events and notes.

1. **Select Notes...**. Selects Note Ons, Note Offs, and Aftertouch. In order for notes to be modified by quantization or randomization, they need to be selected first, otherwise some menu entries are disabled.. Notes can be selected in the piano roll of via this menu. This menu provides two note-selection options:
   - **Select all**, selects all notes in the pattern; The `Ctrl-A` will also select all of the events in the pattern editor.
   - **Invert selection**, which inverts the selection of notes.
2. **Note timing/velocity...**. This menu offers three ways to tweak the timing of the selected notes:
   - **Quantize** quantizes the selected notes in time, the same way as the **Quantize** ("**Q**") button.
   - **Tighten** This operation merely a less strict form of quantization.
   - **Jitter** Jittering modifies the timing of a note by adding or subtracting a small random of time.
   - **Randomize velocity** This operation modifies the velocity of a note by a small amount.
3. **Pitch transpose** allows uniform transpostion regardless of the key and scale in force for the pattern. Selecting this item entry brings up a sub-menu.
4. **Harmonic transpose**, which makes sure that all transpositions stay on the selected scale. If the scale selection is **Off**, this is the same as plain pitch transpose, and so is not shown.
5. **LFO** allows for modulating of control values with a low-frequency oscillator. See section 6.8 "Pattern Editor / Bottom Row" on page 94.
6. **Pattern fix** allows for fixing up a new pattern that was recorded with some timing errors, or transforming the pattern in various ways. See the figure below.

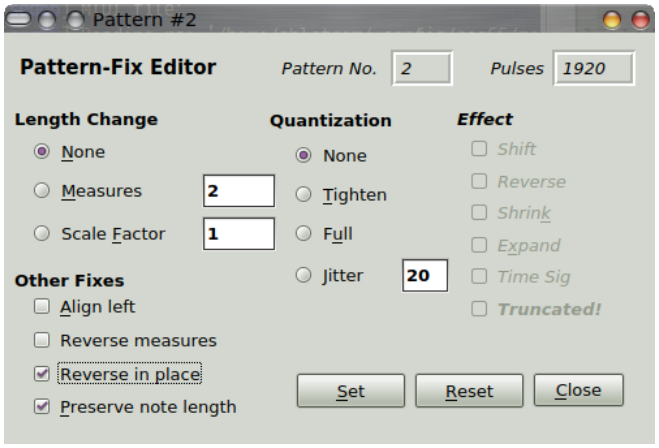


Figure 27: Pattern Fix Dialog

When this dialog is used, the time-stamp of *every event* in the pattern will be changed. It is fairly intuitive to use. Once selected, the pattern number is shown in the title-bar and in a read-only text field. The **Length Change** selector allows for the following actions:

- **None**. No length change will occur. Alignment, reversing, and quantization can still be done.
- **Measures**. Allows the measures (length) of the pattern to be changed in two ways, depending on how the number is entered: as a simple integer (e.g. 1, 1.0, 0.25), or as a simple time-signature fraction (e.g. "3/4").
  - **Measures**. Entering an integer or a floating-point scale factor will scale the pattern events, and change the number of measures, if applicable. If the measure value is less than `1.0`, the pattern will only be compressed. If less than 1 measure in length, the result will be 1 measure; *Seq66* cannot allow less than 1 measure.
  - **Time Signature**. Entering a fraction such as **3**/**4** or **12**/**8** will change the time signature of the pattern **and** *compress it to 1 measure*.
- **Scale Factor**. Scales the pattern. For example, 2.0 doubles the length, and 0.5 halves the length. The **Measures** is changed to be enough to contain the new length, but only if the scale is greater than 1.0. One thing to be aware of is that, if the pattern is expanded, the new measure count depends on the location of the last event in the pattern. For instance, setting the scale factor to 100 might result in 76 measures, not the 100 that would be set if measures was set to 100.

The **Quantization** selector allows for various types of quantization. It works the same as the **Tool** entries of the same name (see section 6.2 "Pattern Editor / Second Row" on page 78), but works on *all events*, not just selected events. Also included is a **Jitter** option, with a range for the jitter. This option slightly randomizes the time-stamps of note events.

Other Fixes provides some minor items, as follows.

- **Align left** allows the pattern to be aligned to the left of the sequence, removing any delays. This is useful when recording a pattern live and not quite hitting the first beat in time.

- **Preserve note length** prevents Note Offs from being scaled, which would shorten or length the notes.
- **Reverse measures** reverses all of the events in the whole pattern, flipping the measures completely. For example, if all the events occur near the end of the pattern, after reversal they will all appear near the beginning of the pattern.
- **Reverse in place** This is probably a more natural reversal method. The relative location of a cluster of notes doesn't change, but the notes are simply reversed in place.

Effect does not do anything except show the effect of the changes. (Making it read-only might make it almost invisible in some themes.) Note the **Truncated!** check-box. If checked, then the length of the pattern has reduced enough to drop some events. This only happens with the time-signature option. The process followed is:

1. Apply the left-alignment (if selected).
2. Modify the number of measures (if selected).
3. Do the scaling (if selected).
4. Perform the quantization/tightening (if selected).

Lastly, the effect of the change is shown, if applicable. Note that changing the measure in this dialog is different from changing the measures in the "length" dropdown, which only changes the measures. Changing the measures to a large number on a pattern of 1 measure will greatly expand the notes!

Any accidental changes can be **Reset** in this dialog. Undo in the pattern-editor itself with `Ctrl-Z` affects only MIDI events, not things like changing time-signature from the drop-downs.

5. Follow Progress. This button toggles whether or not the progress bar follows progress in long patterns. Turning off this feature is useful when one wants to concentrate on the current measure without the paging to subsequent measures that occurs with the "follow progess" feature.

6. Grid Snap. Grid snap selects where the notes will snap when *drawn* and when *moved* (but not when lengthened/shortenend). That is, it selects the snap-spacing for the notes The following values are supported: **1**, **1/2**, **1/4**, **1/8**, **1/16** (*the default value*), **1/32**, **1/64**, and **1/128**. Additional values are also supported: **1/3**, **1/6**, **1/12**, **1/24**, **1/48**, **1/96**, and **1/192**. The button to the left of this control resets it to the default value.

7. Note Length. Note length determines the duration of inserted notes. Like the **Grid Snap** values, the following values are supported: **1**, **1/2**, **1/4**, **1/8**, **1/16** (*the default value*), **1/32**, **1/64**, and **1/128**. Additional values are also supported: **1/3**, **1/6**, **1/12**, **1/24**, **1/48**, **1/96**, and **1/192**. The button to the left of this control resets it to the default value.

8. Zoom. Horizontal zoom is the ratio between MIDI pixels and ticks, written as "pixels:ticks", where "ticks" is the "pulses" in "PPQN". For example, 1:4 = 4 ticks per pixel. Supported values are **1:1**, **1:2** (*the default value*), **1:4**, **1:8**, **1:16**, and **1:32**, along with more values to support higher PPQN tunes: **1:64**, **1:128**, **1:256**, and **1:512**. The default zoom is 2 for the standard PPQN value, 192, but it increases for higher PPQN values, so that the default zoom looks sensible. As the right number (ticks) goes higher, the effect is to zoom out, and show more of the pattern.

9. Key of Sequence. Selects the desired musical key for the pattern. The following keys are supported: **C**, **C#**, **D**, **D#**, **E**, **F**, **F#**, **G**, **G#**, **A**, **A#**, and **B**. Changing the key shifts the

marked note-rows for the **Musical Scale** setting and indicates the base notes of the key in a **bold** font. The small key button resets the key to **C**.

The musical key that a sequence/pattern is set to is saved in the MIDI file along with the rest of the data for the sequence. **However**, a change made to the key, scale, or background sequence (not to be confused with the background-recording sequence) in the pattern editor can be saved in the whole song, so that opening another sequence will apply the same settings to that sequence. This is an optional feature, supported as noted below. Also see **Musical Scale** below for the scale-identification feature.

If the global-sequence feature is enabled, and the user selects a different key, scale, or background sequence in the pattern editor, then *all* patterns share the selected key, scale, or background sequence. Furthermore, these settings are saved in the "proprietary" section of the MIDI file, where they are available for all patterns.

If the global-sequence feature is *not* enabled, and the user selects a different key, scale, or background sequence in the pattern editor, then only that pattern will use the selected key, scale, or background. The key, scale, or background sequence change will be saved in the MIDI file only for that pattern, as a SeqSpec meta event. The global-sequence feature setting can be made in the 'usr' configuration file.

10. Musical Scale. Selects the desired background scale for the pattern; it provides a way for someone to key in notes that are only in that scale. When a scale is selected, the following features are supported:

- The notes that are *not* in the scale are shown as grey in the piano roll.
- For harmonic transposition, the notes are shifted so that they remain in the selected scale.
- The exact notes that are considered "in-scale" shift according to the value of the selected **Key of Sequence**.

The following musical scales are supported so far:

- **Off (Chromatic)**
- **Major (Ionian)**
- **Minor (Aeolian)**
- **Harmonic Minor**
- **Melodic Minor**
- **Whole Tone**
- **Blues**
- **Major Pentatonic**
- **Minor Pentatonic**
- **Phrygian**
- **Enigmatic**

Please let us know of any mistakes in these scales. Note that the **Melodic Minor** scale is supposed to descend in the same way as the natural **Minor** scale, but there is no way to support that trick in *Seq66*.

One can select which **Musical Scale** and **Key** the piece is in nominally, and *Seq66* will grey those keys on the piano-roll that are *not* in the selected scale for the selected key. This is purely visual; a user can still add off-key notes. This feature makes it easier to stay in key while playing and recording. The scale will shift when a different **Key** is selected.

The scale that a pattern is set to is saved in the MIDI file along with the rest of the data for the pattern. A change made to the key, scale, or background pattern in the pattern editor can be saved globally, so that opening another pattern apply the same settings to that pattern. This is a configurable feature in the 'usr' file; see "global_seq_feature". This option allows applying the key/scale/background-sequence either globally (all patterns) or locally (per-pattern), with each pattern holding its key, scale, and background-sequence settings in SeqSpec meta events.

The pattern editor's piano roll has a little secret: the **Scale Identifier**. When the piano roll has focus and `Ctrl-K` is pressed, all of the notes in the pattern are analyzed to try to determine the both the key and the scale of the existing notes. The method is not sophisticated... the notes are counted and are matched against all of the keys (C to B) and scales supported by *Seq66*. The combinations with the highest number of notes are then shown in a message box. This simple analysis depends on having at least 8 notes in the pattern, and it is possible to get weird results if there are only a few *different* notes, as in a simple bass line. Don't expect miracles from this feature. A more sophisticated analysis, the *Krumhansl-Schmuckler* key-finding algorithm, could be used, but it is a bit too complex for our needs, which are basic.

11. Background Sequence. One can select another pattern to draw on the background to help with writing corresponding parts. The button brings up a small menu with values of **Off** and **Set 0** (at a minimum). The 0 is a set number; sets are numbered from 0 to 31. Additional set numbers appear in the menu for each set that has data in it. Under the **Set 0** entry, a menu appears. Once the desired pattern is selected from that list, it appears as dark cyan note bars, along with the normal notes that are part of the pattern.

The background sequence that shows is saved in the MIDI file along with the rest of the data for the sequence/pattern. A change made to the key, scale, or background sequence in the pattern editor is saved in the editor, so that opening another sequence will apply the same settings to that sequence. This is an optional feature, as noted earlier.

12. Chord Generation. One can insert chords with one click. (This feature comes from user "stazed" and his *Seq32* project [36].) Select the desired chord type first. Once a value other than **Off** is selected, drawing mode will add multiple notes representing the chord created, with the clicked note value as the base of the chord.

6.3 Pattern Editor / Measures Ruler

The measures ruler ("bar indicator", or "Time") consists of a *timeline* at the top and the **L marker** and **R marker** items. The following are the elements next to and on the **Time** line:

- **Vertical Zoom Buttons**. These buttons allow the user to compress, reset, or expand the piano roll vertically to a certain degree. (The `v`, `V`, and `0` keys offer the same functions.) A more permanent change in vertical grid scaling can be made by setting the **Editor Key Height** value in the **Edit** / **Preferences** / **Display** tab, or in the 'usr' file.

- **Time Line**. The **Time** (or **Measures**) bar provides an explicit count of beats and bars. It follows the horizontal zoom of the piano roll. It also has a couple tricks, which are shown in the diagram below.
  - **In the upper half** of the time-line, the mouse pointer changes to a vertical pointer. Clicking there then shows a vertical bar and a red dot; these mark the starting position of playback. This is useful for reviewing some notes.
  - **In the lower half** of the time-line, the mouse pointer changes to a "finger" icon. Left-clicking there then moves the "L" marker to that point. Right-clicking there moves the "R" marker to that point. ("R" will never precede "L", though). If the **Loop** button in the main window is active, then playback will loop between the "L" and "R" buttons. This looping now works with both Live and Song modes.

The following figure shows three steps in cursor movement, and the final result for setting a time-signature.

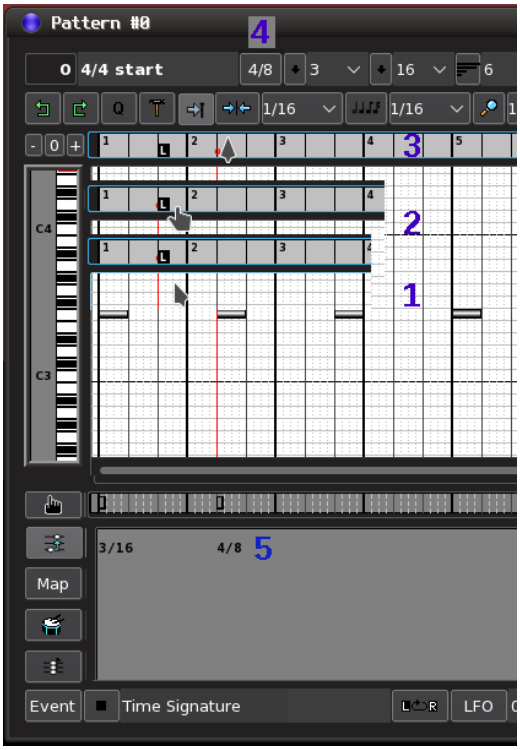


Figure 28: Setting a Time Signature

The steps shown are

1. The mouse cursor is in the piano roll, and has the normal appearance for the current mouse cursor theme.
2. The mouse cursor is in the bottom half of the time line, and here it's a pointing finger. When left-clicked, the "L" marker is set there. When right-clicked, the "R" marker is set there.
3. The mouse cursor is in the top half of the time line, and is shown as a vertical arrow. A left- or right-click sets the current position in the pattern, and a red vertical line marks that position.

4. The time-signature button (the "4/8" shown in the top bar) is clicked, and a time-signature is added at that point.
5. The data pane switches to show Time Signatures. These cannot be edited in the data pane, but in the event bar above it, they can be selected by drawing a box around them, and then be either deleted or moved with the Left/Right arrow keys.

Note that the "L" and "R" markers can be selected via the keyboard using their respective shifted key. Once selected, the marker can be moved left or right using the left and right arrow keys. Also note that the default position of the "R" marker is at the end of the fourth measure, so it might not be visible in the pattern editor without scrolling to it.

6.4 Pattern Editor / Piano Roll

The piano roll is the center of the pattern/loop/track/sequence editor. When a pattern is opened in the editor, the piano roll scrolls automatically to the first notes.

The piano roll is accompanied by a thin "event bar" ("event pane") just below it, and a taller "data pane" or "data area" just below that. While the pattern editor is very similar to note editors in other sequencers, it is a bit different in feel. A good mouse with at least 3 buttons is very helpful for editing. Buttons and keystrokes enhance the ease of editing.

The piano roll shows notes, and, optionally, a background pattern or a scale. Notes are shown as narrow rectanges; the background pattern and scale are shown as bars running the length of the piano roll.

When the piano roll has keyboard focus, the `Space` key starts and stops playback, rewinding to the beginning when stopped. The `.` (period) key starts and pauses playback, without rewinding. The `Esc` key stops playback, exits "paint" mode, and can be enabled to close the pattern editor. This functionality is similar to that of the main window, but these keys are not reconfigurable in the piano roll.

One can page vertically in the piano roll using the `Page Up` and `Page Down` keys. One can go to the leftmost position using the `Ctrl-Home` key, and to the rightmost position using the `Ctrl-End` key, The mouse scroll wheel can also be used to move the panes around. For implementation reasons, the scroll wheel is active *only* in the piano roll.

If no notes are selected, the arrow keys will move the piano row up, down, left, and right in small steps. Otherwise, the selected notes are moved up, down, left, and right.

In addition, the "vi" keys `h`, `j`, `k`, and `l` will act like the arrow keys. This can be convenient, especially if the arrow-keys are unwieldly. For example, the *Microsoft Arc* keyboard puts all four arrows on one button!

If no notes are selected, then `Ctrl-Left-Arrow` and `Ctrl-Right-Arrow` will move the progress bar to the left or right by one snap value.

With the note-step feature, if one paints notes with the mouse, the note position advances with each click. If one paints notes via an external MIDI keyboard, the notes are painted and the note position advances. To preview notes entered via a MIDI device, click the **MIDI Thru** button to activate so that they will be passed to the sound generator or software synthesizer.

6.4.1 Pattern Editor / Piano Roll Items

The center of the pattern editor consists of a time panel at the top, a virtual keyboard at the left, a note grid, a vertical scrollbar, an event panel, and a data panel at the bottom.

1. **Beat**
2. **Measure**
3. **Virtual Piano Keyboard**
4. **Notes**

1. Beat. The light vertical lines represent the beats defined by the configuration for the pattern. The even lighter dotted lines between the beats are useful for snapping notes.

2. Measure. The heavy vertical lines represent the measures (bars) defined by the configuration for the pattern. Also note that the end of the pattern occurs at the end of a measure, and is marked by a blocky **END** marker.

3. Virtual Piano Keyboard. The virtual keyboard is a fairly powerful interface. It shows, by shadowing, which note on the keyboard will be drawn. It can be played with a mouse, using left-clicks, to preview a short motif. Every octave, a note letter and octave number are shown, as in "C4". If there is a difference scale in force, then the letter changes to match, as in "F#5".

A right-click in the virtual keyboard area toggles the display between octave-note letters, MIDI note-numbers, and other views. The following figure shows all views, superimposed for comparison.

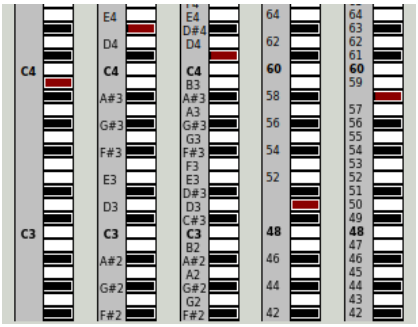


Figure 29: Virtual Keyboard Number and Note Views

4. Notes. Musical notes are indicated in the piano roll by thick horizontal bars with white centers. Each bar provides a visual representation of the pitch of a note and the length of a note. The current scale and background pattern can also be shown in the piano roll.

5. Time Scroll. Allows one to pan through the whole pattern, if it is too long to fit in the window horizontally.

6.4.2 Pattern Editor / Note Painting

When we say "editing" in the context of the piano roll, in part we mean that we will "draw" or "paint" notes. Drawing, modifying, copying, and deleting notes is fairly easy in *Seq66*, though slightly different from other MIDI sequencers.

The *Seq24* note-editing style is as expected for basic actions such as selecting and moving notes using the left mouse button. Drawing a note or event is a bit different, in that one must first enter the drawing mode ("paint mode"). One way is to *click and hold* the right mouse button, and then *click and drag* the mouse to insert notes. (Note that one can use the *Ctrl-left-click* as a middle click.) Another way is to use the `p` key to enter the "paint" mode. To get out of the "paint" mode, press the `x` key while in the sequence editor. The `Esc` key also exits paint mode if the tune is not playing. Also available is a "finger" button (**Note Select/Note Entry**) to click to toggle the mode.

Notes are inserted to be at the current length and grid-snap values for the sequence editor for as long as the buttons are pressed while the mouse is dragged. The length of the note will be that specified in the note-length setting (e.g. "1/16"). This is the "auto-note" feature. The auto-note feature also works with chord-generation. Notes are inserted only up to the specified sequence length. Once notes are inserted, moving the mouse with the left button still held down moves the notes to the new note value of the mouse. If one releases the left button, then presses and holds it again, more notes will be added in the same way. This is a good way to layer notes in a short sequence. The draw mode has the following features:

- Notes are continually added as the mouse is dragged ("auto-notes").
- Notes cannot be added past the "END" marker of the pattern, which marks the **Sequence Length in bars** setting.
- As the mouse is dragged while the left button is held in draw mode, notes are either added, or, if already present at that note-on time, are moved up and down.
- If the draw mode is exited, and entered again, then the original notes will not be altered. Instead, new ones will be added.
- Notes can be added while the pattern is playing, and will be heard the next time the progress bar passes over them.

Drawing/painting can also be done while the sequence is playing, and notes will be added to be played the next time the progress bar crosses them.

6.4.3 Pattern Editor / Note Editing

Once notes are in place, whether by recording or using "paint" mode, the piano roll provides a sophisticated set of note-editing actions.

1. Event Selection. There are various ways to select events and copy, delete, or modify them using the mouse or the keyboard in the piano roll:

- `Ctrl-A`. Pressing the `Ctrl-A` key will select all of the events in the pattern editor.
- `Ctrl-E`. Pressing the `Ctrl-E` key will select all of the events in the pattern editor that have the channel that is selected in the channel dropdown. This selection is useful if one wants to move events from one channel into another pattern.
- `Ctrl-N`. Pressing the `Ctrl-N` key will select all of the notes in the pattern editor that have the channel that is selected in the channel dropdown. This selection is useful if one wants to move notes from one channel into another pattern.

- **Left Click**. Pressing the left button on a note or a event deselects all other notes or events, and selects the item clicked on. The selected note will turn orange (or the configured palette color).
- **Left Click Drag**. Pressing the left mouse button and dragging also lets one select ("lasso") multiple events and notes. The selected notes will turn orange. Adjustments can be made to one or more notes by selecting one or more notes, and then applying one or more special "selection actions" to the selection. Be careful! If you `Ctrl-left-click-drag` on an already-selected note, the drag will change the length of *all of the notes in the selection*.
- **Ctrl Left Click**. Pressing the `Ctrl` key and the left button on a note or an unselected event *adds* that event to the selection.
- **Left Click Drag Selection Up/Down**. To move notes in pitch, once selected, grab one of the notes in the selection and drag it upward or downward. Also, when a selection is in force, the `Up` and `Down` arrow keys will change the pitch of every note in the selection. The smallest unit of pitch change is one MIDI note value.
- **Left Click Drag Selection Left/Right**. To move notes in time, once selected, grab one of the notes in the selection and drag it leftward or rightward. Also, since a selection is in force, the `Left` and `Right` arrow keys can also be used to change the time of every note in the selection. The smallest unit of time change is the **Grid snap** value, which might be a 16th note, for example.
- **Ctrl Left Click Drag**.
  - Pressing the `Ctrl` while left-click-dragging *on unselected events* lets one make additional selections of multiple events and notes.
  - Pressing the `Ctrl` while left-click-dragging *on an already-selected event* lets one stretch or compress the lengths of all notes in the selection. This feature is called *event stretch* or *event compression*. Notes can be shortened below the default note length by event compression. There is currently no way to change the length of the note using a keystroke.
- **Deselect Notes**

The selection, copying, and pasting of notes has some minor tricks to remember. When some notes are selected, the effective selection box goes from the first note to the last note, and from the top-most note to the bottom-most note. When pasting the notes, place the mouse cursor so that it lies on the desired row for the top-most note, and on the desired time location for the left-most note. After pasting, be sure to verify the notes in the new location.

Warning: Reducing or increasing the length of a note selection by too much causes the note or notes to "wrap-around" to the end of the pattern boundary and grow more from the beginning of the sequence. If it happens, one probably ought to undo it.

The **Tools** button described in section 6.2 "Pattern Editor / Second Row" on page 78can also be used to modify selections. Once one or more notes are selected, they can be modified in time, pitch, or length, as described above.

Warning: If one moves the selection too low or too high in pitch, whether with the mouse or the arrow keys, any notes that go below the lowest MIDI pitch or above the highest MIDI pitch **will be lost**! If done using the mouse, the undo feature (`Ctrl-Z`) will work. If done using the arrow keys, the undo feature does not work! Be careful, especially if you have a fast keyboard repeat rate!

Note that there is no possibility of note loss with a change in time. When a note disappears at one end of the pattern boundary, it wraps around to the other end. Cool.

2. Copy/Paste. Copying, cutting, and pasting is supported by selecting a number of events or notes, and using the Cut (`Ctrl-X`), Copy (`Ctrl-C`), and Paste (`Ctrl-V`) keys. When the notes are selected, one can delete them with the `Delete` or `Backspace` key. If the events are *cut*, using the `Ctrl-X` key, then they can be pasted, using the `Ctrl-V` key, then moving the cursor to the desired place, and clicking.

One can move the selection box using the arrow keys, to the desired location, and then click to drop the notes at that location. Selected notes that are cut or copied can also be pasted into *other* pattern editor dialogs; that is, they can be pasted into other sequences.

6.4.4 Pattern Editor / Other Keys

Here are some other, less standard,‘ keys useful in the pattern editor piano roll:

- `c`. Pressing the `c` key will attempt to use the note-mapper data (provided by a `*.drums` file) to change the notes in the pattern. This will work only if the pattern is marked as transposable, to add some safety against multiple pitch changes; these are useful once when converting from one drum machine to General MIDI.
- `Ctrl-D`. Pressing the `Ctrl-D` key will clear the pattern clipboard.
- `f`. Pressing the `f` key will attempt to fix wrap-around notes by moving the note.
- `Ctrl-k`. Pressing the `Ctrl-k` key will analyze all the notes in the pattern to try to guess its scale, as discussed earlier.
- `q`. Pressing the `o` key will toggle recording for the pattern.
- `q`. Pressing the `q` key will quantize the selected notes.
- `r`. Pressing the `r` key will quantize the selected notes.
- `t`. Pressing the `t` key will partially quantize (tighten) the selected notes.
- `u`. Unlinked notes no longer occur. But.... Pressing the `u` key will remove any unlinked notes found in the pattern. This fix is a stop-gap until we can figure out the best way to prevent unlinked notes while handling recording of notes near the end of the pattern length.
- `=`. Pressing the `=` key relinks any unlinked notes found in the pattern. This causes the notes that are unlinked to be linked, and thus wrap around.
- The default keystroke for starting playback is the `Space` character.
- The default keystroke for stopping playback is the `Escape` character.
- The default keystroke for pausing playback is the `Period` character.

6.4.5 Pattern Editor / Zoom Keys

After a left-click in the piano roll, the **z**, **Z**, and **0** can be used to zoom the piano-roll view *horizontally*. The **z** key zooms out (smaller), the **Z** key zooms in (larger), and the **0** key resets the zoom to the default value. The horizontal zoom feature also affects the time-line (measures indicator) and the data area.

The note display can also be zoomed vertically. The **v** key zooms out vertically to make the notes thinner, the **V** key zooms in vertically to make the notes fatter, and the **0** key resets the zoom to

the value of the "key height" setting in the 'usr' configuration file.

6.5 Events Pane

Also known as the "events pane" or "events editor". The narrow (a few pixels high) events strip shows discrete events, such as `Note On` and `Note Off`. The events that are shown are selectable in the **Event** category selector and the "Selector for Existing Data"" drop-downs at bottom left. These and other events appear as small squares in the event strip, along with a black vertical bar in the **Data Pane** with a height proportional to the data-value of the event and a numeric representation of that value. The event value (data) editor (directly under the event strip) is used to change note velocities, channel pressure, control codes, patch select, etc.

*We currently recommend being careful of editing or selecting events in that pane (feel free to disobey), because **more work is needed**.* Note On and Note Off events should not be inserted in the event strip; it is too easy to screw up. In fact, selection and editing is disabled for **Note On**, **Note Off**, and **Aftertouch**.

Other event types (including tempo) can be inserted via the event strip. To do that, first select the kind of event to insert using the **Event** button in the bottom panel. Then place the mouse cursor in the event strip. Right-click to make the drawing cursor appear at the exact spot where the event must go. While holding the right button, click the left button. A small square for the event will appear. Or click-drag to insert a number of events, each at a snap value.

One can also left-click in that section, then hit the `p` key to go into "paint" mode, and hit the `x` key to escape that mode. Note that the "finger" button *does not* apply to the event panel (it applies only to the piano-roll). Should one want more of the same event, continue to hold both buttons and drag the mouse. One event should appear at each beat (16th note) position that is crossed.

Be careful when using smaller snap values (1/32, 1/64, etc.) to insert events. Move the mouse cursor very slowly, otherwise some snap values might be skipped, leaving missing events. In that case, either remove the events and try again, or use the event editor to add events at the missing tick positions.

To move the event(s) to a different time, select it/them via the left button. Then drag the selection left or right as desired. The left and right arrow keys can also be used. it is currently not possible to move them to positions smaller than the beat size; temporarily reduce the beat size if desired. Also, for regularly-spaced events, selected events can be hidden when moved into the next non-selected event. The event values can be edited via the data panel, described in the next section.

6.6 Left Buttons

Once the events are in place, the next step is to modify the data values of the events as needed. But first, note the buttons at the left.

1. **Transpose**
2. **Note Map**
3. **Drum Note Mode**
4. **Chord Generation Reset**

1. Transpose. This button toggles the ability of the sequence to be transposed. If transpose is enabled for that pattern, the button will be highlighted as per the current desktop theme. Patterns for drums should, in general, not be transposable.

2. Note Map. If the pattern is transposable, then this button is enabled. If clicked, it applies the note-mapper to all of the notes in the pattern. See section 12.7 "'drums' File" on page 184. It is most useful for converting percussion from older drum sets to General MIDI drums. Enable transposition, apply the mapping, and then disable transposition to avoid transposing again (e.g. by accident).

3. Drum Note Mode. This button changes from normal note mode to drum note mode. In the drum mode, the notes are drawn as small red diamonds without any duration. They are also entered the same way. This is a feature adopted from *Kepler34*.

4. Chord Generation. This button resets the chord-generation feature to **Off**. It's located by the data pane in order to save space in the first row.

6.7 Data Pane

Now on to the **Data Pane** itself, also known as the "data panel". The events that are shown in this pane are selectable in the **Event** category selector and the "Selector for Existing Data"" drop-downs at bottom left. **Modify Event Data** offers a way to alter the event data values. Many different events can be altered in the data pane: Note On and Note Off velocities, program changes, aftertouch, channel pressure, pitch wheel, and tempo. Text events are also displayed (useful in karaoke), but they cannot be edited in this pane (instead see the **Session** tab's **Song Info** controls and the **Event Editor**.).

The events values for the currently selected category of events are shown in this window as vertical lines of a height proportional to the value, topped by a circular grab handle. The exceptions are program changes and tempo, which are shown by small circles, yellow in the case of tempo. Also, the range of tempos in the data panel is set to match the `usr!bpm-minimum` and `usr!bpm-maximum` settings in the 'usr' file. This range is for display purposes. See section 12.4.6 "'usr' File / User MIDI Settings" on page 160.

These values can be easily modified by left-click-dragging the mouse past each line, to level it off at the given value. Easier to try it than explain it. Right-click-drag also works the same. When notes are *selected*, and the mouse is used to change the values (heights) of the lines in the event-data area, *only the events that are selected* are changed. The data-values of *unselected* events are left unchanged. A cool feature from *Seq24*.

Also, as the mouse passes over the data pane, events near the mouse acquire a grab handle, which can be used to select the event and modify its value by moving the mouse up or down.

Note that some events are shown as small circles instead of a line; each circle has the numeric value next to it.

Tempo events are shown as a small yellow circle. This circle can be grabbed and moved up or down.

Figure 30: Data Pane Tempo

A new feature is the ability to add tempo events in the data pane. To add a string of tempo events, first select **Tempo** from the **Event** dropdown. Move to the data pane, press and hold `Ctrl` and then left-click-drag the mouse to form a line as shown here.

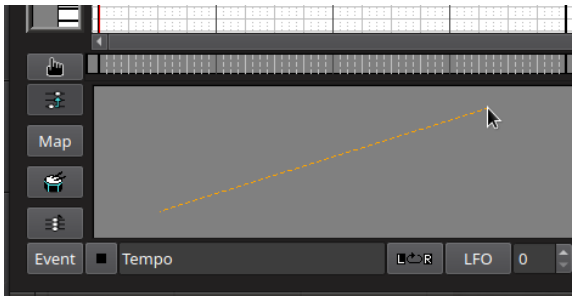


Figure 31: Data Pane Tempo Line Draw

Note that there are no events shown yet. While still holding `Ctrl`, release the left mouse button, and the tempo events appear. Release the `Ctrl` key. If this line of tempos is fine, right-click to lock them.

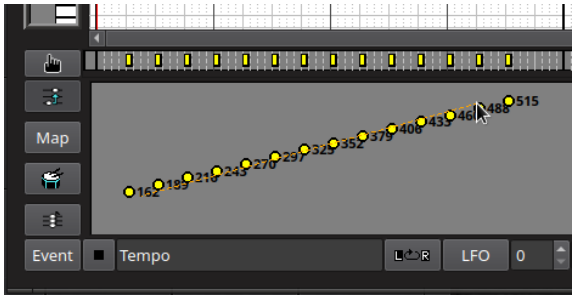


Figure 32: Data Pane Tempo Events Drawn

Otherwise, move the mouse around to change the tempo events, as shown here. Use the right-click to exit this mode.

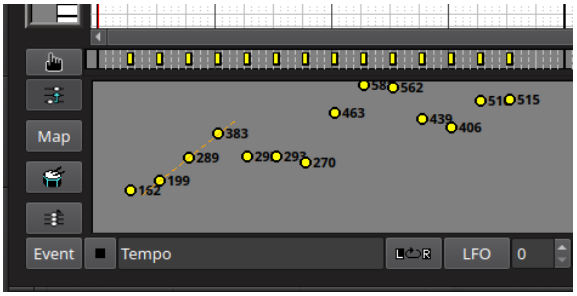


Figure 33: Data Pane Tempo Events Altered

Again, note that the range of the tempo events is determined by the BPM minimum and maximum values specified in the 'usr' file. There is currently no **Preferences** setting for these values.

Also note that, to delete tempo events in the pattern editor, select the events in the thin event pane, then press either `Delete` or `Ctrl-X`. They can also be deleted (slowly) in the event editor.

Program Change events are shown as a small open circle with a numeric value. Both Tempo and Program Change event values can be modified by dragging a line as discussed above. (We're still working on select-and-drag for these events.)

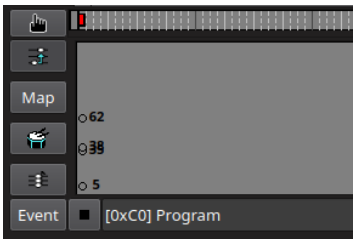


Figure 34: Data Pane Program Change

As of version 0.99.9, we have added *Seq32*-style grab-handles to the Note events in the data panel. They are shown only if an event is selected, or if the mouse is on top of the event line. As the mouse moves through the data panel, the grab-handles are shown as circles at the top of each line. If the line is clicked, it is selected. If Ctrl-clicked, the selected line is added to the selection. If an empty spot in the data pane is clicked, all events are unselected. When present, the circular grab-handle can be clicked-and-held and moved up and down to change the event value.

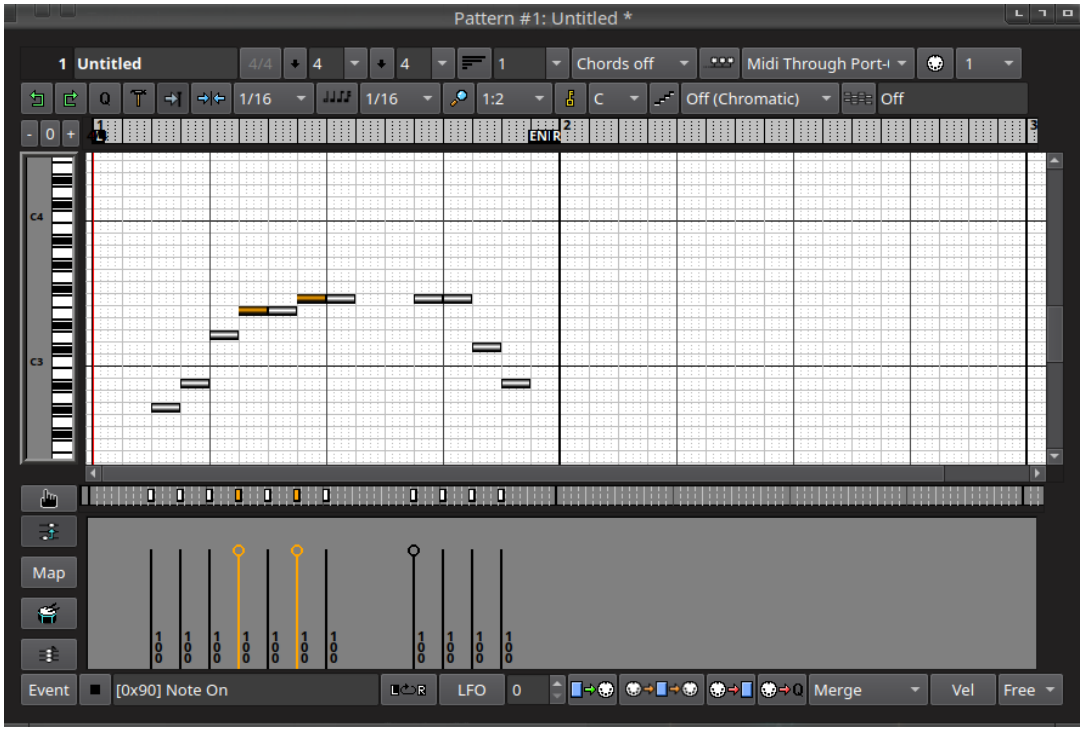


Figure 35: Data Pane Note Grab-Handles

Any events that are selected in the piano roll or event strip can have their values modified with the mouse wheel. Data values can also be modified using the **LFO** pane (see below).

6.8 Pattern Editor / Bottom Row

The bottom row of the pattern editor provides for selecting events for viewing and editing, MIDI playback, pass-through, and recording.

1. **Event Category Selector**
2. **Selector for Existing Data**
3. **Selected Event Name**
4. **LFO Panel**
5. **Live Loop Count**
6. **Armed/Muted Toggle** (Data To MIDI Buss)
7. **MIDI Thru Toggle**
8. **MIDI Record Toggle**
9. **MIDI Record Quantized**
10. **Loop Record Type** (Merge, Replace, Expand, One-shot)
11. **Record Velocity and Reset**

1. Event Category Selector. This button brings up the following context menu, so that the user can select the category of events to view and edit.

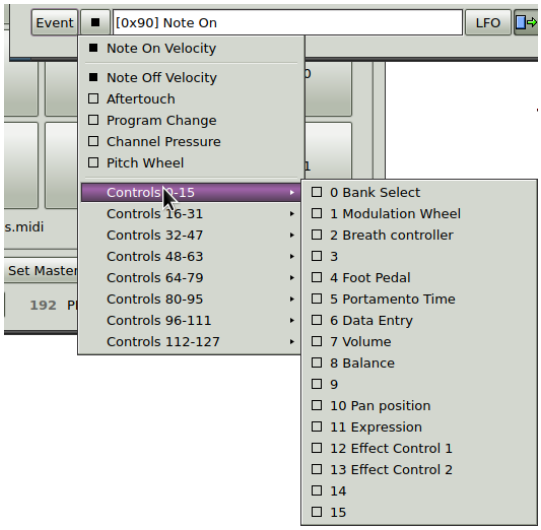


Figure 36: Pattern Editor Event Button Context Menu

Note the squares. Some might be filled (black), most are empty. Filled squares indicate that the sequence has some events of that type. Otherwise, there are no such events in the sequence. Useful in deciding if it is worth selecting the event.

The sub-menus of this context menu show 128 MIDI controller messages. They also use the squares to indicate if there are any events of the type shown in the menu. These sub-menus can be modified by editing the 'usr' file:

```
$HOME/.config/seq66/seq66.usr
```

to make it match one's instrument.

2. Existing Event Menu. The existing-event selector is a small button (with a black-square icon) that brings up a menu with only existing events shown. Unlike the event-selector described above, this menu shows only the actual events existing in the track, for quicker selection.

3. Event Selection. Shows the selection event, with its event number shown in hexadecimal notation, and the name of the event shown.

4. LFO Panel. An LFO (low-frequency oscillator) allows data events to be modulated by some rudimentary wave functions. By clicking on the **LFO** button or using the `Ctrl-L` key, the following window appears, with a set of 4 vertical sliders:

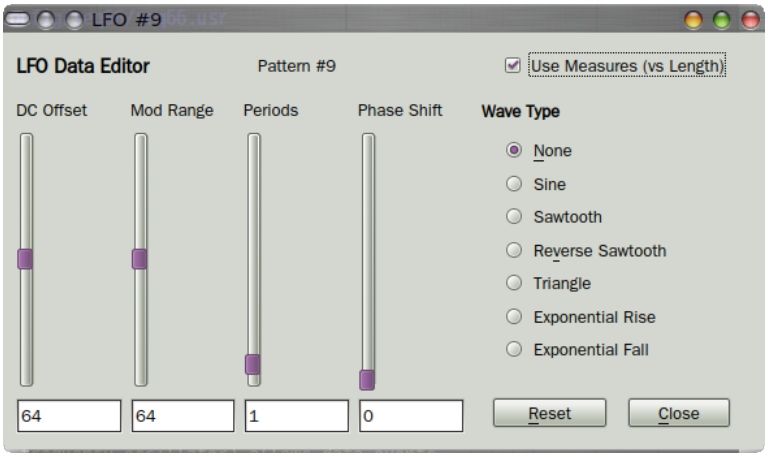


Figure 37: Pattern Editor LFO

1. **Use Measures (vs Length)**. The length of a waveform period can be determined by the length of the pattern (2 measures as shown next to the label "LFO Data Editor") or by the length of a measure. This check-box determines that. In the diagram, it is checked, so the 2 measures covers four periods of the sine wave. Especially useful in modifying long patterns.
2. **DC Offset** (**D**). Provides a kind of DC offset for the data value. Starts at 64, and ranges from 0 to 127. In the diagram above, one can see that the 0-value line for the sine wave is at 64.
3. **Depth** (**R**). Controls the depth of modulation. Starts at 64, and ranges from 1 to 127. The data values range from $y = DC - R$ to $y = DC + R$. For the sine-wave shown above, the range is 0 to 128 (actually 127).
4. **Periods**. Indicates the number of periods per pattern length. For long patterns, this parameter should be set high, to even show an effect. It is also subject to an 'anti-aliasing' effect, especially for short patterns. Try it!
5. **Phase Shift** (**P**). Provides the phase shift within a period of the LFO wave. A value of 1 is a phase shift of 360 degrees (one whole period). Thus, the data at $P = 0$ would look exactly the same at phase $P = 1$.
6. **Wave Type**. Selects the kind of wave to use for the LFO:
    1. **None**. This setting is useful if one wants to change only the DC offset.
    2. **Sine**. Modulates via a sine wave. Change the **Phase Shift** to get a cosine function. If the **DC Offset** is below 64, then negative values are converted to positive. As an example, setting the DC offset to 0 will produce the absolute value of the `sin()` function.
    3. **Sawtooth**. Provides a ramp modulation.
    4. **Reverse Sawtooth**. Provides a ramp modulation in the opposite direction.
    5. **Triangle**. Modulates via a triangle wave; somewhat similar to a sine wave.
    6. **Exponential Rise**. Provides an exponential ramp modulation. This ramping is easiest to see if the DC offset is 0, and the Mod range is 127. Varying the **Phase Shift** with the slider will move the peak of the curve left to right.
    7. **Exponential Fall**. Provides a ramp modulation in the opposite direction.

    Note that one might have to change a parameter slightly to see the effect of the new waveform.
7. **Reset Data**. This button restores the initial pattern event data. Useful when one applies modulation that one ultimately does not like.
8. **Close**. Closes the LFO panel.

In addition to the **Reset Data** button, Ctrl-Z can be applied multiple times to undo changes one at a time. Every motion of a control causes a complete change.

5. Live Loop Count. Normally, in Live mode, a pattern plays endlessly if left alone. If this counter is set to a value greater than 0, then the pattern will loop only that number of times in Live mode. For example, if set to 1, then the pattern acts like a "one-shot" loop. This can save having to use queuing quickly to handle an intro phrase. To loop *endlessly*, set this value to 0. Also set it to 0 when playing the MIDI tune in **Song** mode. Otherwise, weird behavior will be observed.

6. Armed/Mute Toggle. This button causes the pattern to be output to the selected MIDI output buss, which will normally be connected to a software or hardware synthesizer, to be heard. This item performs muting/unmuting (disarming/arming) in the same way a pressing the corresponding pattern button in the **Live** frame.

7. MIDI Thru Toggle. This button routes incoming MIDI data through *Seq66*, which then writes it to the MIDI output buss. When a new pattern editor is opened, and the new-editor-editor settings (section 12.4.8.3 "'usr' File / Additional Options / [new-pattern-editor]" on page 163) are false, one can click the **Thru** button first to redirect MIDI controller input to the synthesizer port, and have it be heard, without arming the pattern or turning on MIDI Record. Note, though, that if MIDI Record is toggled on and off, the Thru function is effectively disabled. To restore it, toggle the Thru off, then on, again. Also note that Thru will remain enabled when the pattern editor is closed.

8. MIDI Record Toggle. This button routes incoming MIDI data into *Seq66*, which then saves the data to its buffer, and also displays the new information (notes) in the piano roll view. Note that *Seq66*'s pattern grid can be put in various recording modes (e.g. overdub/merge versus overwrite) where, instead of muting/unmuting the patterns, it turns on recording (without opening the pattern editor).

9. MIDI Record Quantized. This button will causes MIDI data to be recorded, but be quantized on the fly before recording it. The quantization is to the current snap value. Quantization can be turned on globally in *Seq66*'s pattern grid as well.

10. Loop Record Type. In *Seq24*, the pattern recording worked by merging new notes played as the pattern to be recorded was looped. This method allows a loop to be built up bit-by-bit. *Seq66* adds two more methods from Stazed's *Seq32* project. The three methods are:

1. **Merge**. This is the normal style of recording loops, where notes can accumulate as the loop repeats.
2. **Overwrite.** When the loop starts over, and a note is pressed, then the existing notes in that loop are erased, and the new note is added. This provides a good way of correcting major mistakes, live. It will not work if adding notes while not recording. This mode can cause incomplete notes if one holds the note and releases it in the next iteration, leaving a partially-drawn note behind. The workaround is to try again.
3. **Expand**. Expansion operates when playing and recording. Once the end of the loop is near, when notes are being input, another measure is added to the length of the loop. This continues indefinitely, whenever any notes are being played and recorded. This works best in **Live** mode. In **Song** mode, only one measure is added each time play is started, and playback stops when the song is complete. A workaround is to add a long track, perhaps empty, to the song. Awaiting user complaints to decide what to do.

4. **Oneshot**. When this option is set, with the record button on, and no pattern playing, recording won't start until a note comes in, and when the first note comes in, the progress bar starts at the left (time 0). As each new set of notes at the same timestamp come in, the notes are recorded and the current time advances by one snap value. The length of each note is determined by the snap size for the spacing of drawn events. At the end of the pattern, recording stops automatically. See the figure below for a recording from a *Yamaha DD-11*.

Remember that *Oneshot playback* is a kind of auto-step/step-edit recording, and that auto-step/step-edit can also be done by click-dragging the mouse. Do not confuse it with *Oneshot playback*.

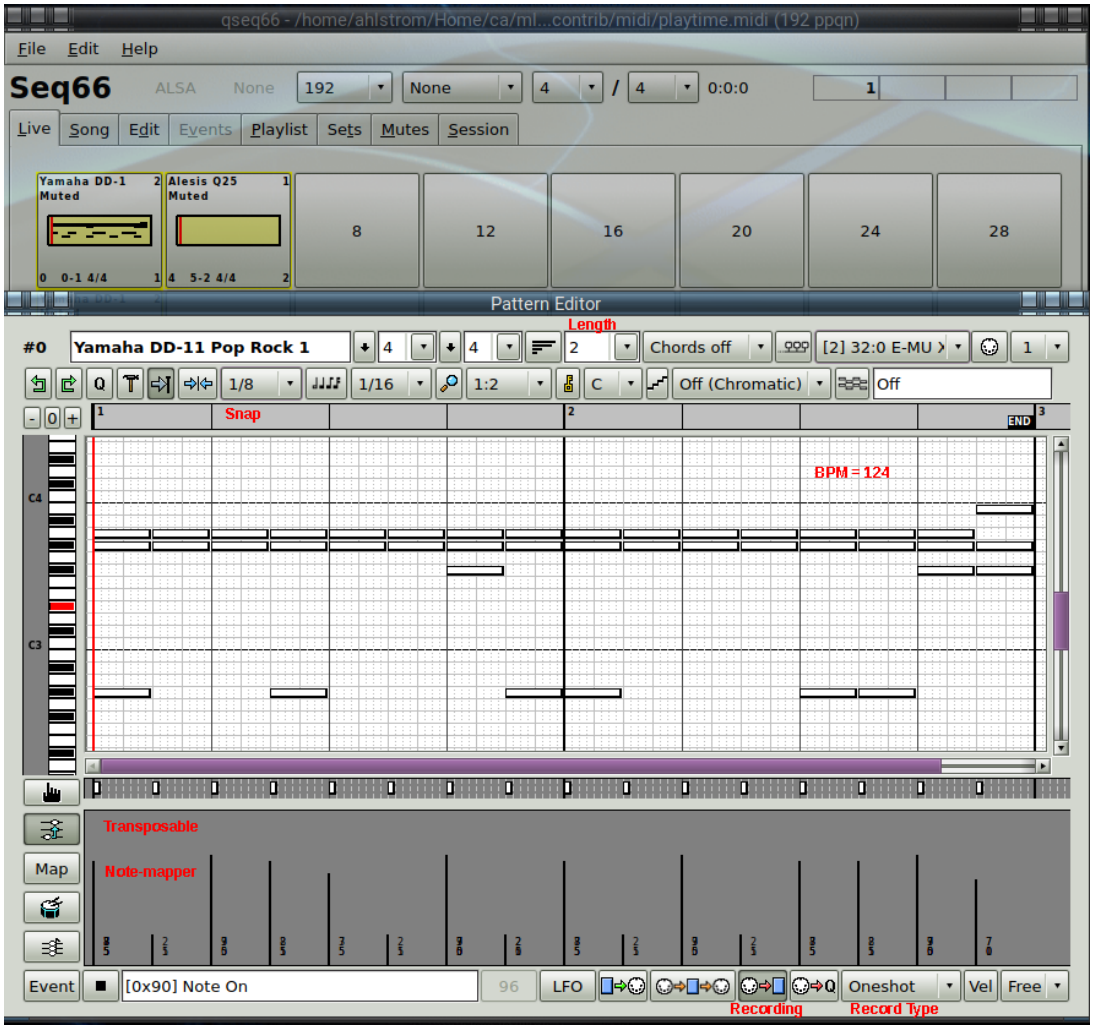


Figure 38: One-Shot Pattern Recording

In the figure above, we set up to record input from the port attached to a *Yamaha DD-11* drum machine. After some trial and error, we set:

- **Length** to 2 measures (see the red text in the figure).
- **Snap** to 1/8th, which sets the spacing of drawn notes as well. This determines the position of each new note.

- **Note Length** to 1/16th, which sets the length of drawn notes as well.
- **Recording Type** to **Oneshot**.
- **Recording** on. This is the button with the MIDI DIN connector pins going into the blue vertical rectangle.

Also, the **BPM** was set to 124 in the main window, to match the "39" tempo in the DD-11, which maps to 124 bpm. Finally, we picked the *Pop Rock 1* style. Once this setup was in place, clicking the DD-11 Start/Stop button started recording automatically at time 0, and it stopped automatically at the length/end of the pattern.

Why is the snap used instead of the note-length? Because we're using the auto-step (step-edit) recording feature... the snap determines where the next note begins, and the length determines the length of the note to create. However, the note-length is a property of the piano roll, not the pattern itself. The pattern uses the snap-length during auto-step/step-edit. This kind of note recording can be done in two ways:

1. Left-click-drag the mouse.
2. Turn on recording without the pattern playing, then enter notes; recording stops when the end of the length is reached.

Now, the DD-11 is an old instrument from the pre-General-MIDI days. So, in order to play back this pattern on something like *QSynth* or *Hydrogen*, we need to re-map the notes to GM drum notes. In the 'rc' file, the proper note-mapper file is specified:

```
[note-mapper]
"GM_DD-11.drums"
```

We copy the recorded pattern and paste it into another slot for safety. We click the **Transposable** button for that pattern to enable the **Map** button. Then we click the **Map** button, and the notes shift (not shown). We click the **Transposable** button again to disable transposing, and save the file. Playing it into channel 10 of *QSynth* shows that it sounds a lot like the original DD-11 drums.

11. Vol. This button resets the volume (velocity) of note recording to the **Free** setting. See the next item.

12. Velocity. This dropdown allows setting the volume of the recording to either the incoming velocity or to the specified velocity. The velocity values are shown at the right side of each menu entry. These values correspond to MIDI volume levels from 127 down to 16. If the **Free** item is selected, then the incoming note velocity is preserved.

6.9 Pattern Editor / Common Actions

This section is a catch-all for actions not described above.

6.9.1 Pattern Editor / Common Actions / Scrolling

We still need to work out whether or not to use the scroll wheel. in Seq66, as we need to keep multiple event panes in sync while scrolling. Let us describe the actions that can be performed with a scroll wheel, or with the scrolling features of multi-touch touchpads. There are three major scrolling actions available when using mouse scrolling, with the mouse hovering in the piano-roll area:

- **Vertical Panning (Notes Panning)** Using the vertical scroll action of a mouse or touchpad moves the view of the sequence/pattern notes up and down. One can also click in the piano roll, and then use the `Page-Up` and `Page-Down` keys to move the view up and down in pitch.
- **Horizontal Panning (Timeline Panning)** Holding the Shift key, and then using the vertical scroll action of a mouse or touchpad moves the view of the sequence/pattern time forward and backward. One can also click in the piano roll, and then use the `Shift Page-Up` and `Shift Page-Down` keys to move the view left and right in time.
- **Horizontal Zoom (Timeline Zoom)** Holding the Ctrl key, and then using the vertical scroll action of a mouse or touchpad zooms the view of the sequence/pattern time to compress it or expand it. One can also click in the piano roll, and then use the `z` , `Z` , and `0` keys to change the timeline zoom.
- **Vertical Zoom (Notes Zoom)** Additional buttons for vertical zoom have been added: **-**, **0**, and +. One can also click in the piano roll, and then use the `v` , `V` , and `0` keys to change the notes zoom. The zoom can make the note-rows large enough to use on a touch screen.

The actions of this scrolling are smooth and fast. If an event is selected in the piano-roll area or the (thin) event area, then the scrolling increases or decreases the value of the event. In the case of a note, this increases or decreases the velocity of the note. For all events, this increases or decreases the length of the vertical line that represents the value of the event.

6.9.2 Pattern Editor / Common Actions / Close

There is no **Close** button in the pattern editor. One can use window-manager actions, such as clicking on the **X** button of the window frame, or pressing the exit key defined in the window manager.

7 Song Editor

The **Song Editor** is also known as the "arrangement panel" or "performance editor". The **Song Editor** combines all patterns into a complete tune with controlled repetitions of each pattern. It shows one row per pattern/loop/sequence, with the placement of each pattern at various and possibly multiple time locations in the song. In *Seq24* parlance, the song editor creates a *performance*, and the performance is implemented by a set of triggers. Triggers are internal timing items stored with each pattern when a *Seq66* MIDI tune is saved. In **Song** mode, these triggers, not the user, control playback.

Tip In the installed `data/midi` directory, there are sample files for the tunes "Europe Endless" and "Peter Gunn" that illustrate what can be done with the song editor. They are accompanied by descriptive text files. Be sure to check them out.

Two song editor windows can be brought onscreen, one in the **Song** tab, and one in an external window. The **Song** tab and a **Song** window can be shown at the same time.

Once playback is started in the song editor (using the `Space` or `.` keys), live mode is disabled. The song editor takes over the arming/unarming (unmuting/muting) shown in the patterns panel. The highlighting of armed/unarmed patterns changes according to whether the pattern is triggered in the song editor. If one tries to change the muting in the patterns panel, the song editor immediately returns the pattern to the state it has in the song editor. The only way to manually change the muting then is to click the pattern's label in the song editor. Both the song editor and the patterns panel both reflect the change in muting in the user-interface.

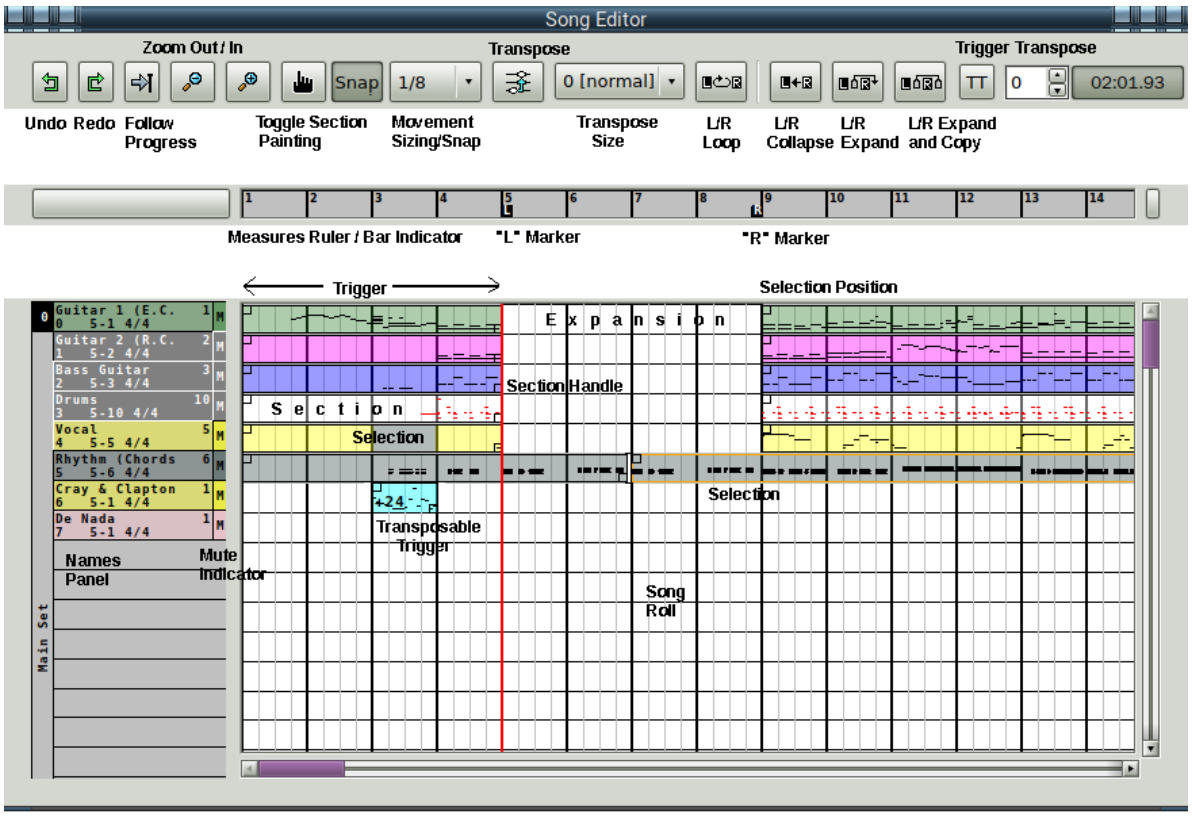


Figure 39: Song Editor Window, Annotated

Note the major items shown:

1. **Top Panel** (recent updates not yet shown)
2. **Measures Ruler**
3. **Patterns (Names) Panel**
4. **Song Roll**
5. **Bottom Panel**

The estimated duration of the tune appears at the right of the top panel. It is calculated from the

length of patterns and the song triggers that may be present. Here are some of the features for the song editor:

- Toggling of the mute state of multiple patterns via the name fields of the patterns.
- Optional pattern coloring (selected in the Patterns panel)
- A configurable progress bar.
- **Undo** and **Redo** buttons.
- A **Transpose** button and transposition drop-down selector.
- Red coloring of events for patterns that are not transposable, such as drum tracks.
- Horizontal zoom (and a simple vertical zoom) via buttons and keystrokes.
- Opening a pattern editor via a double-click on any occupied track.
- Creating a pattern via a double-click on any unoccupied track.

The song editor is a bit complex; for exposition, we break it into sections, starting with playback keystrokes.

7.1 Song Editor / Playback Keystrokes

The song roll (center panel) of the song editor provides keystrokes for starting, stopping, and pausing playback. Other keystrokes are described in section 7.5.2 "Song Editor / Song Roll / Keystrokes" on page 107.

The keystroke for starting playback is the `Space` character. It starts the playback of the song at the **L marker**. The **L marker** serves as the start position for playback in the song editor. One can change the start position only when the performance is not playing. The `Space` character also toggles playback. This character also rewinds the song to the beginning when stopping.

Another keystroke for stopping playback is the `Escape` character. This character also rewinds the song to the beginning when stopping.

The keystroke for pausing playback is the `Period` character. When used, it keep the progress at the same spot as before.

Note that there are no stop, pause, and play buttons in the song editor. They are supplied by the main window. The **Song** tab can be activated in the main window to keep the playback buttons nearby.

7.2 Song Editor / Top Panel

The top panel shown earlier provides quick access to actions and configuration.

1. **Undo**
2. **Redo**
3. **Follow Progress**
4. **Zoom Out and Zoom In**
5. **Toggle Section Painting**
6. **Record Snap**

7. **Grid Snap**
8. **Transpose**
9. **L/R Loop**
10. **L/R Collapse**
11. **L/R Expand**
12. **L/R Expand and Copy**
13. **Expand Song Roll** (not shown in figure)
14. **Trigger Transpose**

1. Undo. The **Undo** button rolls back the last change in the layout of a pattern. Each time it is clicked, the most recent change is undone. Also implemented via `Ctrl-Z`.

2. Redo. The **Redo** button reapplies the last change undone by the **Undo** button. Also implemented via `Shift-Ctrl-Z`.

3. Zoom Out and Zoom In. These buttons change the horizontal zoom. Zoom can also be changed via the keystrokes `z`, `0`, and `Z`. Vertical zoom is also supported, by buttons to the left of the time-line and by keystrokes, as discussed below.

4. Follow Progress. **Follow Progress** toggles the mode of following progress for longer songs. WHen active, the song roll pages right to keep up with the progress bar.

5. Toggle Section Painting. **Toggle Section Painting** toggles the ability to drag the mouse along the pattern's timeline to create triggers to indicate when the pattern plays. Short patterns will be duplicated one or more times as the mouse is dragged. This mode can alsp be changed via the keystrokes `p` and `x`.

6. Grid/Record Snap. The **Movement Sizing/Snap** setting, if enabled, allows not only full clips of a pattern to be added, but smaller intervals listed below. It turns on record-snap for recording live performance triggers. It also enables the grid snap functionality, which ndicates the horizontal grid snap for movement actions and trigger drawing. If disabled, it allows the trigger to be placed and to be smoothly extended in either direction, without snapping, when the mouse is moved left or right. It allows exact recording of the musician's arming/muting of patterns. Unlike the **Grid Snap** of the pattern editor, the units of the song editor snap value are in fractions of a measure length. The following values are supported:

1/1, 1/2, 1/4, 1/8, 1/16, and 1/32

Note that arming/muting can be done in the "names" panel using a *right-click* on the pattern name. Also note that changes that occur within the snap value will cause odd recording, so be sure to set the snap value low enough.

When creating triggers for patterns longer than a measure, the pattern may wrap, so that beginning notes appear at the end of the trigger, and notes can wrap around. To avoid this, a trick is needed:

1. Undo that trigger insertion.
2. Select the `Length` Snap value.
3. Insert the trigger.
4. Click another snap value.
5. Drag the trigger, usually to the left, until it reaches the desired snap location.
6. Verify that the whole pattern is in place with the notes exactly placed as in the pattern.

This trick is annoying, and we're not sure if note wrap-around is a feature or a bug.

7. Transpose. **Transpose** consists of two controls: a combo-box to select the transpose direction and amount, and a **TT** button to reset the transposition value to 0. Transposition ranges from -60 to +60, or five octaves either way. Transpostion is applied by setting the value, and then doing a `Shift-left-click` on each trigger that needs that transposition value.

8. L/R Loop. This button appears in the song editor only when it is open as an external window. This button is present in the main window, and so is not needed in the Song tab. For a description of this button, see section 3.5.6 "L/R Loop" on page 26.

9. L/R Collapse. This button collapses the song between the **L marker** and the **R marker**. What this means is that, if there is song material (patterns) before the **L marker** and after the **R marker**, and the **Collapse** button is pressed, any song material between the L and R markers is erased, and the song material after the **R marker** is moved leftward to the **L marker**. Collapsing occurs in all tracks present in the song editor.

10. L/R Expand. This button expands the song between the **L marker** and the **R marker**. It inserts blank space between these markers, moving the song material that is after the **R marker** to the right by the duration of the blank space. Expansion occurs in all tracks present in the song editor.

11. L/R Expand and Copy. This button expands the song between the **L marker** and the **R marker** much like the **Expand** button. However, it also copies the original data that is present after the **R marker**, and pastes it into the newly-available space between the L and R markers.

12. Expand Song Roll. Sometimes one might come to the end of the song and still want to add more triggers. Clicking this button expands the song roll to the right, adding more room for triggers.

13. Trigger Transpose. This button and spin-box support making trigger selections or segments transposable during play-back. This feature is very useful for patterns that repeat many times, but are shifted in pitch at various points. The transposition value ranges from -60 to 0 to +60, in units of semitones. The button resets the value to 0. To apply a transposition value, first set it in the spin-box. Then carefully *shift-left-click* on the desired segment(s) to transpose. A number with a plus-or-minus will appear at the left of the segment to indicate a non-zero transposition. The transposition value will be saved with the trigger when the song is saved.

7.3 Song Editor / Measures Ruler

The measures ruler ("bar indicator", or "Time") consists of a *timeline* at the top and the **L marker** and **R marker** mentioned above. The *measures ruler* is the ruled and numbered section at the top of the arrangement panel. It provides a place to put the left and right markers. In the *Seq24* documentation, it is called the "bar indicator". There are some hidden details (tricks) in the measures panel.

- **In the upper half** of the time-line, the mouse pointer changes to a vertical pointer. Clicking there then shows a red dot; these mark the starting position of playback. This is useful for review.

- **In the lower half** of the time-line, the mouse pointer changes to a "finger" icon. *Left-clicking* there then moves the **L** marker to that point. *Right-clicking* there moves the **R** marker to that point. (**R** will never precede **L**, though). If the **Loop** button in the main window is active, then playback will loop between the **L** and **R** buttons. This looping now works with both Live and Song modes.

Left-click in the bottom-half of the measures ruler to move and drop an **L marker** (**L anchor**) on the measures ruler. *Right-click* in the bottom-half of the measures ruler to drop an **L marker** (**R anchor**) on the measures ruler.

These markers denote the time interval from the left of the **L** marker to the right of the **R** marker. Once these markers are in place, one can then use the *Collapse* and *Expand* buttons to modify the placement of the pattern events. Note that the **L marker** serves as the start position for playback in the song editor even when not looping. One can change the start position only when the performance is not playing.

Another way to move the **L** and **R** markers has been added. To select which marker will move, click the upper half of the time strip (otherwise, the **L** will move, prematurely) to give it keyboard focus. Then press the lower-case `l` key or the lower-case `r` key. *There is no visual feedback that one is in the movement mode.* Then press the `Left-Arrow` or `Right-Arrow` key to move the selected marker. Also included at the same level as the measures ruler are the buttons **-**, **0**, and +, which are used for vertical zoom, as described below.

7.4 Song Editor / Patterns (Names) Panel

The patterns panel is at the left of the song roll. Here are the items to note in the pattern-names panel:

1. **Number**. The number of the screen set.
2. **Title**. The title is the name of the pattern, for easy reference.
3. **Color**. If a color is provided for the pattern, then that color is shown. Also, as an editing aid, the pattern over which the mouse is hovering is shown in a brighter version of the color.
4. **Channel**. The channel number appears (redundantly) at the right of the title.
5. **Buss-Channel**. This pair of numbers shows the MIDI buss number used in the pattern and the channel used for the pattern.
6. **Beat/Measure**. This pair of numbers is the standard time-signature of the pattern.
7. **Trigger Count**. This number shows the number of triggers present in the pattern.
8. **Mute Indicator**. The letter **M** is in a grey box if the track/pattern is muted via song playback, and a white (or colored) box if it is unmuted in song playback. *Left-clicking* on the **M** (or the name of the pattern) mutes/unmutes the pattern. Song muting is effected via a *left-click* on the pattern name. This is also useful in song-recording (where the triggers are recorded). If the Shift key is held while *left-clicking* on the M or the pattern name, then the mute/unmute state of every other active pattern is toggled. This feature is useful for isolating a single track or pattern. Normally, one records song triggers using the grid buttons or MIDI control to turn patterns on and off. One can also *right-click* on the pattern name during song-record and thereby see the trigger(s) being created.
9. **Empty Track**. Completely empty tracks (no track events or meta events) are indicated by a dark-gray filling in the pattern column. Tracks that have only meta information, but no

playable event, are indicated by a yellow filling in the pattern column.

The patterns column shows a list of all of the patterns that have been created in the current song. Each pattern in this list has a track of pattern layouts associated with it in the piano roll section.

Left-clicking on the pattern name or the **M** toggles the muting (arming) status of the track. It does the same thing if the `Ctrl` key is held at the same time.

Shift-left-clicking on the pattern name or the **M** button toggles the muting (arming) status of *all other tracks* except the track that was selected. This action is useful for quickly listening to a single sequence in isoloation.

Double-clicking on the pattern name will either create a new pattern or open up the corresponding pattern window. This feature saves having to move to the live grid.

7.5 Song Editor / Song Roll

The "Song Roll" section of the arrangement panel is where patterns or subsections are inserted, deleted, shrunk, lengthened, or moved. Actions can be done via the mouse or keyboard.

7.5.1 Song Editor / Song Roll / Layout

The song/piano roll provides one line (track) per pattern. It provides another way (besides the live grid) to control and lay out patterns. Patterns can be set up with multiple triggers and can be brought up for editing. Here are features to note in the annotated piano roll area:

1. **Note Events**. Note in the pattern are shown as thin horizontal lines. They can be edited in the pattern editor's piano roll, and in the event editor.
2. **Tempo Events**. Tempo events in the pattern are shown as very small squares. They can be edited in the pattern editor's data panel, and in the event editor.
3. **Pattern Access**. (Since version 0.99.10). Double-clicking (if enabled in the 'rc' file) on a track representing an existing pattern will bring up an external pattern editor windows. Double-clicking on an empty track will create a new pattern and bring up its pattern editor. This new functionality is merely for convenience.
4. **Trigger Creation**. By click-dragging the mouse on a track, in paint mode, a series of triggers can be created; they indicate where the track will be unmuted and playing. See below for more information about triggers.
5. **Selection**. Clicking inside a trigger selects it. Selection is denoted by an orange rectangle around the trigger and a dark grey color in the trigger. A pattern subsection can be moved by the mouse and deleted by keystrokes.
6. **De-selection**. *Left-clicking* or *right-clicking* in an empty area of the song roll will deselect the selection.
7. **Selection Movement**. If one grabs (*left-click*) inside the pattern or pattern subsection, that item can be moved horizontally, as long as there is room.
8. **Section Length ("handle")**. The small squares in two corners of the patterns are the section "handles". By grabbing a handle with a *left-click*, the handle can be moved horizontally to either lengthen or shorten the pattern to the nearest snap position, if there is room to move in the desired direction.

9. **Pattern Subsectioning**. A *middle-click* (or *ctrl-left-click*) inside a pattern inserts a selection position marker in it, breaking the pattern into two equal pieces. We call each piece a *pattern subsection*. This division can be done over and over. There are also options for splitting at the nearest snap point.
10. **Expansion**. Originally, all the long patterns of this sample song were continuous. But, by setting the L and R markers, and using the **Expand** button, we opened up some silent space in the song, just to be able to show it off.

The *Seq24* help files refer to work in the song editor as the "Performance Editor" or "Performance Mode". Adding a pattern in this window is a bit like adding a note in the pattern editor. One clicks, holds, and drags the mouse to insert a copy or copies of the pattern associated with the row in which one is dragging. The longer one drags, the more copies of the pattern that are inserted.

Right-click on the arrangement panel (roll) to enter paint mode, and hold the button. Paint mode does not work while the sequence is playing. Another way to turn on painting is to make sure that the performance editor piano roll has the keyboard focus by *left-clicking* in it, then press the `p` or `i` key to enter the paint/insert mode, and `x` (or `Esc` if not playing) to escape it. See section 7.5.2 "Song Editor / Song Roll / Keystrokes" on page 107.

The song editor supports horizontal zoom in the piano roll. This feature is accessible via the "magnifying glass" buttons, and also accessible via the keystrokes **z**, **0**, and **Z**. The zoom feature also modifies the time-line.

The song editor supports limited vertical zoom in the piano roll. This feature is accessible via the **-**, **0**, and + buttons, and also accessible via the keystrokes **v**, **0**, and **V**.

A *left-click* with a simultaneous *right-click-hold* inserts one copy of the pattern. The inserted pattern shows up as a box with a tiny representation of the notes visible inside. Some patterns can be less than a measure in length, resulting in a tiny box. To keep adding more copies of the pattern, continue to hold both buttons and drag the mouse rightward.

Middle-click (or *ctrl-left-click*) on a trigger in a pattern row to splits the trigger into two triggers. This splits the pattern into two equal *pattern subsections*. Each *middle-click* on the pattern adds a new selection position, halving the size of the subsections as more pattern subsections are added. The `allow_snap_split` option in the 'rc' file allows the split to be made at the nearest snap point instead of in the middle.

When a pattern or a pattern subsection is *left-clicked* in the piano roll, it is marked with a dark gray filling. It can then be moved horizontally if there is room, or be deleted or copied for later pasting.

When a *right-left-click* action is done in this gray area, the result is to *delete* that pattern section or subsection. One can also hit the `Delete` key.

Double-click on one of the trigger bars will either create a new pattern or open up the corresponding pattern window. This feature saves having to move to the live grid.

7.5.2 Song Editor / Song Roll / Keystrokes

There are a number of useful keystrokes in the song roll that can be used once is has focus, by clicking in it.

- Enter "paint" mode. The `p` key enters paint mode, where additional triggers can be added by click-dragging on a pattern row. The `x` key leaves this mode. The "finger" button and the mouse cursor both indicate the status.
- Start/Pause button functionality. When the song roll has keyboard focus, the `Space` key starts and stops playback, rewinding to the beginning when stopped. The `.` (period) key starts and pauses playback, without rewinding. This functionality is similar to that of the main window, but these keys are not reconfigurable in the song roll.
- Undo / Redo / Cut / Copy / Paste of a selected section. Provided by buttons and by these keystrokes:
  - `Ctrl-Z`. Undo.
  - `Shift-Ctrl-Z`. Redo.
  - `Ctrl-X`. Cut. Removes the selection. Can also be done with the `Delete` and `Backspace` keys. The deletion can be undone.
  - `Ctrl-C`. Copy. Copies the trigger for later usage.
  - `Ctrl-V`. Paste. Puts the roll into paste mode. When inserted, each insert goes immediately after the current item or the previous insertion. The same can be done for whole patterns.
- Horizontal (Time) Zoom. These keystrokes work similarly to the pattern editor's piano roll. Provided by buttons and by these keystrokes: `Z`. Zoom in horizontally (i.e. in time). `z`. Zoom out horizontally. `0`. Reset zoom both horizontally and vertically.
- Vertical (Time) Zoom. These keystrokes work similarly to the pattern editor's piano roll. However, there are only three levels of vertical zoom: half-size, normal, and double-size. `V`. Zoom in vertically to get a better view of the patterns and a large grab handle. `v`. Zoom out vertically to see more tracks. `0`. Reset zoom both horizontally and vertically.
- Scrolling. The arrow keys will move the piano row up, down, left, and right. In addition, the "vi" keys `h`, `j`, `k`, and `l` will act like the arrow keys. The mouse scroll wheel can also be used to move the panes around. For implementation reasons, the scroll wheel is active *only* in the piano roll.
- Paging. One can page up and down vertically in the arrangement panel using the `Page Up` and `Page Down` keys. One can page left and right horizontally in the arrangement panel using the `Up-Arrow` and `Down-Arrow` keys.

7.6 Song Editor / Bottom Panel

The bottom panel is simple, consisting of a stock horizontal scroll bar.

8 Event Editor

The *Seq66* **Event Editor** tab is used to view and edit, in detail, the events present in a loop / sequence / pattern / track. It is accessed by right-clicking on a pattern in the **Live** frame, then selecting the **Edit pattern in tab** menu entry. The default keystroke combination for this action is to use the *minus* key followed by the desired pattern's hot-key.

The event editor is not very sophisticated. It is a basic editor for simple edits, viewing, and trouble-shooting. It is disabled if recording a pattern, to avoid refresh issues while recording.

Viewing and scrolling work; editing, deleting, and inserting events work. But there are many possible interactions between event links (Note Off events linked to Note On events, for example), performance triggers, and the pattern, performance, and event editor dialogs. Surely some bugs still lurk. If anything bad happens, do *not* press the **Save to Sequence** button! If the application aborts, let us know! Here are the major "issues":

1. It requires the user to know the details about MIDI events and data values.
2. For safety, it does not detect any changes made to the sequence in the pattern editor; we might add a refresh button.
3. It does not have an undo function. Just don't save!
4. It cannot mark more than one event for deletion or modification. However, if one note event is deleted, the corrsponding linked note event is also deleted.
5. There is no support for dragging and dropping of events.

The event editor is a good way to see the events in a sequence, and to delete or modify problematic events. Additionally, it can be used to add **Set Tempo** and other meta events. If an event is added that has a time-stamp beyond the current length of the sequence, then the length of the sequence is extended. Unlike the event pane in the pattern editor, the event-editor dialog shows all types of events at once.

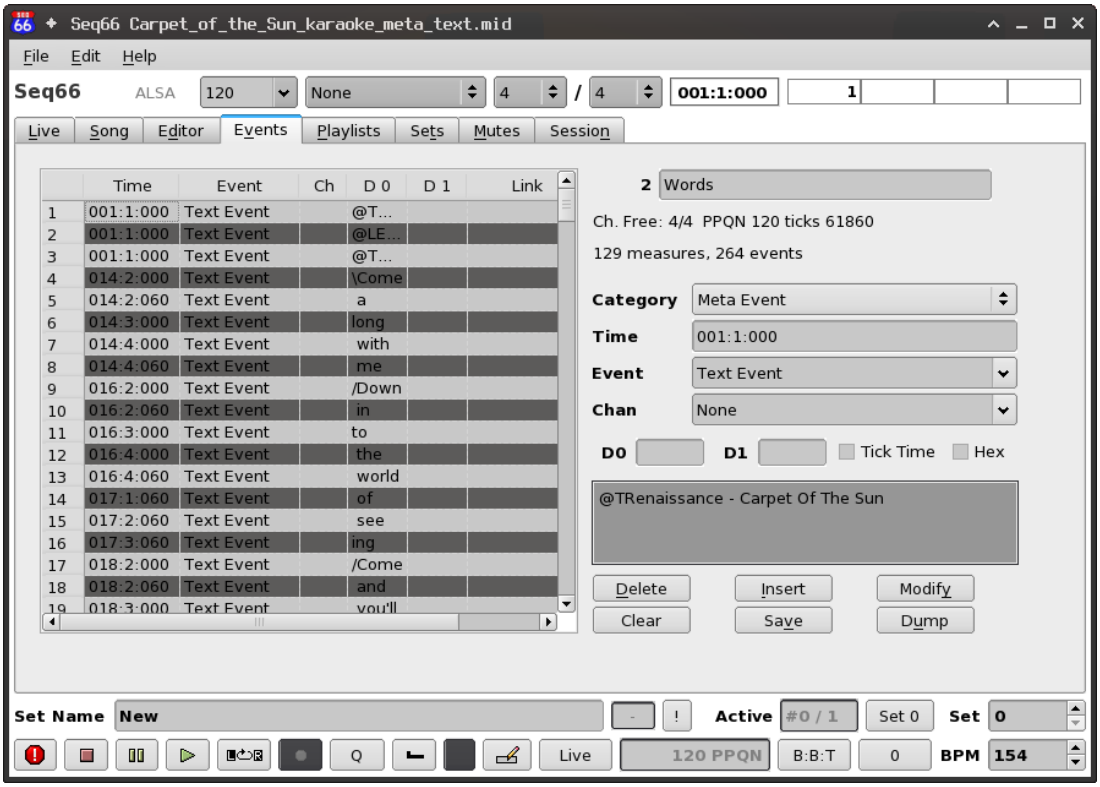


Figure 40: Event Editor Window

The event-editor dialog is fairly complex. For exposition, we break it down into a few sections:

1. **Event Frame**

2. **Info Panel**
3. **Edit Fields**
4. **Bottom Buttons**

The event frame is a list of events, which can be traversed, and edited. The fields in the right panel show the name of the pattern containing the events and other information about the pattern. The edit fields provide text fields for viewing and entering information about the current event, and buttons to delete, insert, and modify events. The bottom buttons allow changes to be saved and the editor to be closed. The following sections described these items in detail.

8.1 Event Editor / Event Frame

The event frame is the event-list shown on the left side of the event editor. It is accompanied by a vertical scroll-bar, for moving one line or one page at a time. Mouse or touchpad scrolling can be used to move up and down in the event list. This movement is even easier than reaching for the scrollbars. Depending on the window manager theme, the currently-selected event is highlighted. (We have been trying to get this table to auto-stretch vertically when the main window is vertically maximized, but have not succeeded so far. Qt!)

8.1.1 Event Frame / Data Items

The event frame shows a list of numbered events, one per line. The currently-selected event is shown in the edit fields. Here is an example of the data line for a MIDI event:

```
17   003:3:128 Note On   Ch 3    0x45:69 0x6b:107 003:4:96
```

This line consists of the following parts:

1. **Index Number**
2. **Time Stamp**
3. **Event Name**
4. **Channel Number**
5. **Data Bytes**
6. **Link**

1. Index Number. Displays the index number of the event. This number is purely for reference, and is not part of the event. Events in the pattern are numbered from 1 to the number of events in the pattern.

2. Time Stamp. Displays the time stamp of the event, which indicates the cumulative time of the event in the pattern. It is displayed in the format of "measure:beat:divisions". The measure values start from 1, and range up to the number of measures in the pattern. The beat values start from 1, and range up to the number of beats in the measure. The division values range from 0 up to one less than the PPQN (pulses per quarter note) value for the whole song. As a shortcut, one can use the dollar sign ("$") to represent PPQN-1. If the **Tick Time** box is checked, then the timestamps are shown in units of "ticks" (MIDI pulses, sometimes called "divisions").

3. Event Name. Displays the name of the event. The event name indicates what kind of MIDI event it is. See section 8.3 "Event Editor / Edit Fields" on page 112.

4. Channel Number. Shows the channel number (for channel-events only) re 1, not 0. (For the user, MIDI channels always range from 1 to 16. Internally, they range from 0 to 15.)

5. Data Bytes. Shows the one or two data bytes for the event. The byte is shown in two formats, hexadecimal and decimal, as in `0x6b:107` ("hex:dec"). For text events, a small portion of the text is shown.

Note Off, Note On, and Aftertouch events requires a byte for the key (0 to 127) and a byte for the velocity (also 0 to 127). Control Change events require a control code and a value for that control code. Pitch wheel events require two bytes to encode the full range of pitch changes. Program change events require only a byte value to pick the patch or program (instrument) to be used for the sequence. The Channel Pressure event requires only a one-byte value. Tempo requires a number (e.g. "120.3") to be typed in.

6. Links. Note events are linked together; each Note On is linked to the corresponding Note Off, and vice versa.

8.1.2 Event Frame / Navigation

Moving about in the event frame is straightforward, but has some wrinkles to note. Navigation with the mouse is done by moving to the desired event and clicking on it. The event becomes highlighted, and its data items are shown in the "info panel". There is no support for dragging and dropping events in the event frame. There is no support for selecting multiple events.

The scrollbar can be used to move within the frame, either by one line at a time, or by a page at a time. A page is defined as one frame's worth of lines, minus 5 lines, for some overlap in paging.

Navigation with keystrokes is also supported, for the Up and Down arrows and the Page-Up and Page-Down keys. Note that using the Up and Down arrows by holding them down for awhile causes autorepeat to kick in. Use the scrollbar or page keys to move through multiple pages. Home and End also work.

8.2 Event Editor / Info Panel

The "info panel" is a read-only list of properties on the top right of the event editor. It serves to remind the used of the pattern being edited and some characteristics of the pattern and the whole song. It also includes one button. Five items are shown:

1. **Sel Linked**. (Not yet shown) If this button is checked, then selecting a Note On or Off event also selects the opposite, linked note. This allow for easy deletion of a complete Note pair.
2. **Sequence Number and Name**. A bit redundant, as the window caption or the pattern also shows the pattern name. It can be set here or in the pattern editor.
3. **Time Signature**. A pattern property, shown only as a reminder. It can be set in the pattern editor.
4. **PPQN** Shows the "parts per quarter note", or resolution of the whole song. The default

PPQN of *Seq66* is 192.

5. **Sequence Channel** In *Seq66*, the channel number is a property of the pattern. All channel events in the pattern get routed to the same channel, even if somehow the event itself specifies a different channel.
6. **Sequence Count** Displays the current number of events in the pattern. This number changes as events are inserted or deleted.

8.3 Event Editor / Edit Fields

The edit fields show the values of the currently-selected event. They allow changing an event, adding a new event, or deleting the currently-selected event.

1. **Event Category** (bold-faced, read-only)
2. **Time** (event timestamp)
3. **Event** (event name)
4. **D0** (data byte 1)
5. **D1** (data byte 2)
6. **Show data in hex format** (not shown in figure)

Important: changes made in the event editor are *not written* to the sequence until the **Save** button is clicked. If one messes up an edit field, just click on the event again; all the fields will be filled in again. That's as much "undo" as the event-editor offers at this time, other than closing without saving.

1. Event Category. Displays the event category of the event in bold-face. Currently, only channel events and a couple of meta events, can be handled, but someday we hope to handle the wide array of system events, and perhap even system-exclusive events.

2. Time. Displays the timestamp of the event. Currently only the "measure:beat:division" format is fully supported. We allow editing (but not display) of the timestamp in pulse (divisions) format and "hour:minute:second.fraction" format, but there are bugs to work out.

If one wants to delete or modify an event, this field does not need to be modified. If this field is modified, and the **Modify** button is pressed, then the event will be moved. This field can locate a new event at a specific time. If the time is not in the current frame, the frame will move to the location of the new event and make it the current event.

3. Category. This drop-down displays the category of the currently selected event:

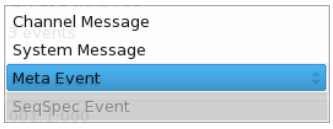


Figure 41: Event Category List

There are four types of messages handled by *Seq66*: Channel, System, Meta, and SeqSpec. Seqspec messages are stored when a pattern is saved, but they are not part of the pattern's event list. Hence this item is grayed out; these message values are shown in various parts of the user interface. See

the table in section 25.2 "Sequencer-Specific Meta-Events Format" on page 251.

4. Time. Shows the time of the event in the format of "measure:beat:divisions" (only). This field can be edited to change the time of the event.

5. Event. Displays the name of the event, and allows entry of an event name. The event name indicates what kind of MIDI event it is. The following event names are supported for Channel events:

1. **Note Off**
2. **Note On**
3. **Aftertouch**
4. **Control Change**
5. **Program Change**
6. **Channel Pressure**
7. **Pitch Wheel**
8. **Tempo**

Selecting one of these names from the dropdown changes the kind of event if the event is modified. Abbreviations and case-insensitivity can be used to reduce the effort of typing. Also, if **Control Change** or **Program Change** are selected, then a data drop-down box is available to select either the controller or the instrument patch (program), and it can fill in the data values. In the future, we may make these configurable, with the control-change following the settings in the 'usr' file, and the program-change being made from a (new) 'patch' file. Currently the programs are General MIDI.

If the **Meta Event** category is selected, the following events are shown:

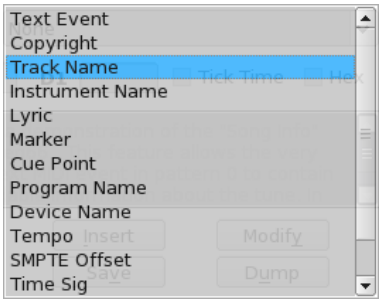


Figure 42: Meta Event List

If the **System Message** category is selected, the following events are shown:

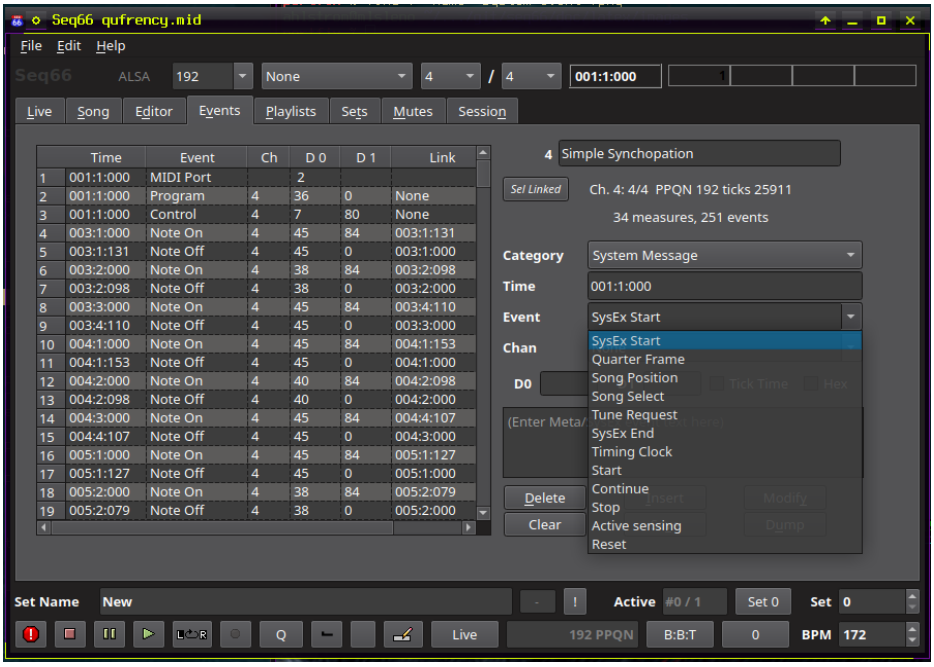


Figure 43: System Message List

At present, most of these message are not editable, but at least they can be seen.

6. Channel. This dropdown shows the channel of the current event, ranging from 1 to 16. If the event is not a channel message, then **None** is shown.

7. D0. Allows modification of the first data byte of the event. One must know what one is doing. The scanning of the digits is very simple: start with the first digit, and convert until a non-digit is encountered. The data-byte value can be entered in decimal notation, or, if prepended with "0x", in hexadecimal notation.

For a Tempo setting only this field is used; Data Byte 2 is ignored. Enter a Tempo value, such as "120", and then click **Insert**. The value is converted to the 3 bytes of a tempo event, and then added at the given timestamp. (Screen refresh is not perfect yet, but reloading the pattern shows the correct tempo.)

8. D1. Allows modification of the second data byte of the event (if applicable to the event). One must know what one is doing. The scanning of the digits is as noted above.

9. Tick time. Selecting this check-box shows the time-stamps in the frame in tick (pulses) format.

10. Hex. Selecting this check-box shows all number items in hex format.

11. Text data. This unlabelled pane shows the text of a meta text event. This field can be edited to add or change a meta text event. The events supported are: *Text Event*, *Copyright*, *Instrument Name*, *Lyric*, *Program Name*, and *Device Name*.

8.4 Event Editor / Bottom Buttons

These are the buttons that act on the edit fields or current event selection:

1. **Delete** (selected event)
2. **Insert** (new event)
3. **Modify** (selected event)
4. **Clear** (all events)
5. **Save** (back to pattern)
6. **Dump** (dump the events to a file)

Note that *Seq66* does not support using the `Delete` and `Insert` keys to supplement the buttons; the `Delete` key is needed for editing the event data fields.

1. Insert. Inserts a new event, described by the **Event Timestamp**, **Event Name**, **Data Byte 1**, and **Data Byte 2** fields. The new event is placed in the appropriate location for the given timestamp. If the timestamp is at a time that is not visible in the frame, the frame moves to show the new event, so be careful.

2. Modify. Deletes the current event, and inserts the modified event, which is placed in the appropriate location for the given timestamp. (This feature does not work with linked Note Ons and Note Offs).

3. Clear. Deletes all of the events in the event table. As with all edits, does not become official until the **Save** button is clicked.

4. Save. Saves all of the events in the event table into the original sequence. There is no way to undo this action. This button does not close the dialog; further editing can be performed. The Save button is enabled only if some unsaved changes to the events exist. Any sequence/pattern editor that is open should be reflected in the pattern editor once this button is pressed.

5. Dump. Write the events to a text file in the same directory as the MIDI file, very useful for troubleshooting. The name of the file is of the form:

```
midi_file_name-pattern-#.text
```

where '#' is the pattern number. For example, if the loaded file is

`/home/user/miditunes/The_Wild_Bull.midi`

and the pattern is 9, then the resulting dump-file is

`/home/user/miditunes/The_Wild_Bull-pattern-9.text`.

Good luck with this tab. Bug reports are appreciated.

9 Session Management

A session is a group of applications and their configuration and connections. Session management recreate complex setups and provides some uniformity of application control in a session. The first thing to do for session management is to make sure that the application is capable of various levels of session management, from *UNIX* signals to a complete session manager like the *Non/New Session Manager*. Basic session management consist of being able to properly start the application and let it run properly during its life-cyle, whether it is a command-line application or a graphical application. *Seq66* supports session management in three ways:

1. **Signals**. During a normal run, *Seq66* will respond to signals to save and to quit. The normal configuration files and command-line options will be used, and can be marked to be saved at exit. This mode is useful with *nsm-proxy*, a way to script applications that don't have *NSM* support.
2. **JACK Session** Deprecated, but implemented nonetheless. Many still use it. This session manager allows the configuration files to be stored in a separate directory, for *Seq66* to be started, and files to be saved. No restrictions on where the MIDI files can be stored.
3. **Non Session Manager** Known as *NSM*, and in the form of a fork of that project, *New Session Manager*, it provides a replacement for *JACK Session*. It requires all files to be stored in a session directory, and provides commands for saving, quitting, hiding/showing the user-interface, and more. Like *JACK Session*, it allows control over the startup of multiple applications, the process of saving a session, and provides a way to save their patching (connections) in *JACK*. However, it supports more functionality and has strict requirements the application must follow. Development of NSM has, for various reasons, been suspended, but offshoots such as *Agordejo* ([3]) and *RaySession* ([31]), which are front ends for the *New/Non Session Manager* ([23]), continue to advance.

For session management, *NSM* is the way to go. *JACK* session management is provided for those who still use it. There are other session solutions, such as *aj-snapshot*, *Claudia*, and *Chino*. For now, we do not discuss them.

The desired session can be set in the **Edit** / **Preferences** / **Session** tab. But note that, if started by *NSM*, *Seq66* will detect it and nonetheless set up for NSM usage. The *NSM* setting is useful for attaching to a pre-existing known session. *JACK* session management events are processed only if *JACK* is selected. *JACK* session management will still start *Seq66* in an existing session, if *JACK* is not selected.

Also note that sometimes one will want the session manager to make the JACK connections. In this case, go to **Edit** / **Preferences** / **JACK** / **Jack Auto-Connect**, uncheck that option, and restart *Seq66*. This option, `jack-auto-connect` can also be changed in the 'rc' file. The *NSM* tool called *jackpatch* can also be used to manage connections.

9.1 Session Management / Signals

By default, the basic form of session management in *Seq66* occurs by signals. A session manager can start *Seq66*, and it can tell *Seq66* to save or stop. Starting is done by a system call to spawn the application. The save and stop actions are supported by sending the following signals to the application:

- `SIGINT`. This signal stops *Seq66*. It corresponds to using `Ctrl-C` from the command-line to stop *Seq66*. This signal should work for both the graphical and command-line application. As *Seq66* shuts down, it does its normal saving of the current state of the configuration.
- `SIGTERM`. This signal also stops *Seq66*. It can be sent by an application to exit *Seq66*.
- `SIGUSR1`. This signal tells *Seq66* to save. This action will save the current MIDI file.

One application that can control *Seq66* via these signals, when not in session mode, is *nsm-proxy*:

https://non.tuxfamily.org/wiki/nsm-proxy

NSM-Proxy is a simple *NSM* client for wrapping non-NSM capable programs. It enables the use of programs supporting LADISH Level 0 and 1, and programs which accept their configuration via command-line arguments. There is a command-line version and a graphical version. More to come on how to use `nsm-proxy`.

9.2 Session Management / JACK Session

Although deprecated by the *JACK* authors in favor of *NSM*, we are implementing *JACK* session (JS) management for the benefit of people who either do not know of *NSM* or do not want to implement or use it.

Seq66, as a JS-aware applications, is set up to

1. Register with a JS manager.
2. Respond to messages from the JS manager.
3. Be startable with session information.

A response to a JS message will do one of the following:

- Save the application's state into a file, where the directory is supplied by the session manager.
- Reply to the session manager with a command-line that starts the application, with information to restore its state, such as the name of the file holding its state information.

JS-aware clients identify themselves to the session manager by a UUID (unique universal identifier). The session manager provides it to the client application as an integer represented as a string. This can be passed to the session manager when registering, but *Seq66* just uses the value given to it (for now).

For this discussion, we will use the *JACK* session implementation in the *QjackCtl* application. Also, read the script stored in `seq66/data/linux/startqjack` to set up *QjackCtl* to run *JACK* and kick off *a2jmidid*; it should be added to the *QjackCtl* configuration.

Once that setup is made (installing the script and configuring `qjackctl`, then start `qjackctl`. Verify that there are a number of system audio and MIDI playback and capture port, *PulseAudio JACK* sinks and sources if the system uses *PulseAudio*, and that there are "a2j" MIDI ports for all of your USB hardware devices.

Then start *Qsynth* so it uses *jack* for MIDI and *jack* (or *pulseaudio*) for audio. Then run *Seq66* with *JACK* for slave transport and for MIDI (either command works the same):

```
$ qseq66 --jack-slave --jack-midi
$ qseq66 --jack-slave --jack
```

Load a file, make sure its MIDI output goes to "fluidsynth" or "qsynth", and plays. In your desired location (e.g. `/.config/seq66/sessions`, create a new session directory (e.g. `qtest`).

In *qjackctl*, open the **Sessions** dialog. Click **Save**, and choose the directory just created. In the dialog should appear entries for MIDI capture and playback for "fluidsynth" and "seq66", all the "a2j" USB devices, plus an entry for *JACK* client *seq66master* or *seq66slave* that shows something like:

```
qseq66 --jack --jack-master --jack-session-uuid 84670 --home ${SESSION_DIR}
qseq66 --jack --jack-slave --jack-session-uuid 84670 --home ${SESSION_DIR}
```

In the **Connections** dialog of *QjackCtl*, all of these ports will be shown in the MIDI tab, auto-connected appropriately. In the sessions directory that was created, will be seen an empty `seq66master` or `seq66slave` directory, and a `sessions.xml` configuration file containing the information shown in the sessions dialog. Exit *Seq66*, *QSynth* and *QjackCtl* (in that order).

One issue is that *QSynth* does not support *JACK Session*. Try it with *Yoshimi*, which does support it.

9.3 Seq66 Session Management / NSM

The *Non Session Manager* is an API implementation for session management for Linux audio/MIDI. *NSM* clients use a well-defined *OSC* protocol ([24]) to communicate with the session management daemon.

Note that *Non Session Manager* is in a state of suspended development, and has been reimplemented as a *GitHub* project, the *New Session Manager*.

The applications it manages should be installed normally (that is, for system-wide usage, in `/usr/bin/` or `/usr/local/bin`).

9.3.1 Session Management / NSM / First Run Without NSM

This section discusses what happens when *Seq66* is installed, then run outside of any session from the console or an application menu. For a discussion where *Seq66* is run for the first time under *NSM*, see section 9.3.2 "Seq66 Session Management / NSM / Run in NSM" on page 120.

Generally, after installing *Seq66*, or when creating a new setup (such as a play-list) it is good to run it normally first, to simplify trouble-shooting. This action creates the configuration files in the default location, `/home/user/.config/seq66`:

```
$ qseq66
No 'rc' file, will create: qseq66.rc/ctrl/midi/mutes
No 'usr' file, will create: /home/user/.config/seq66/qseq66.usr
File exists: /home/user/.config/seq66/qseq66.rc
Saving initial config files to session directory!
Writing 'rc': /home/user/.config/seq66/qseq66.rc
Writing 'ctrl': /home/user/.config/seq66/qseq66.ctrl
Writing 'mutes': /home/user/.config/seq66/qseq66.mutes
Writing 'usr': /home/user/.config/seq66/qseq66.usr
. . .
```

Then exit *Seq66* to ensure the configuration files are created. Optionally, in this initial setup, one can also create a 'playlist' file and a 'drums' file, or copy them from:

```
/usr/share/seq66-0.91/data/samples
```

to

```
/home/user/.config/seq66
```

and modify them appropriately. Another first-time modification to consider is setting up *Seq66* to use the *JACK* audio/MIDI subsystem (on *Linux*). In the 'rc' file, look for the following line:

```
[jack-transport]
jack-midi = false
```

And change it to:

```
[jack-transport]
jack-midi = true
```

Another first-time modification to consider is using virtual ports (option `-manual-ports`) versus the automatic port connections *Seq66* normally makes. This setup allows the user to manually make connections between *Seq66* and other MIDI applications. In the 'rc' file, look for the following lines:

```
[manual-ports]
virtual-ports = false   # 'true' = manual (virtual) ALSA or JACK ports
output-port-count = 8   # number of manual/virtual output ports
input-port-count = 4    # number of manual/virtual input ports
```

And change the virtual-ports line to:

```
[manual-ports]
virtual-ports = true    # 'true' = manual (virtual) ALSA or JACK ports
```

It is then important to start `qseq66` in the normal manner again, and verify that everything works as expected.

We have added a command, **File / Import / Import Project** command to import the configuration files from one directory into the current *NSM* session configuration directory. This import used to be automatic, but is too surprising to an unsuspecting user.

9.3.2 Seq66 Session Management / NSM / Run in NSM

Note: When *Seq66* is run in *NSM* for the first time, a stock default configuration is saved when *Seq66* exits. This is different from earlier behavior, where the home configuration was imported automatically. Now, the user must use the **File / Import / Import Project...** command.

For illustration, we run *NSM* from a terminal window, which can be very helpful when problems occur.

```
$ non-session-manager
[non-session-manager] Starting daemon...
[nsmd] Session root is: /home/user/NSM Sessions
NSM_URL=osc.udp://mycomputer.mls:19625/
[nsmd] Listing sessions
```

If *NSM* refuses to start, make sure that the `liblo` library from the OSC project is installed. If it is installed, then check the `/etc/hosts` file to make sure that a loopback interface is defined. In some versions of *Linux*, it isn't defined properly, and the *NSM* daemon (`nsmd`) will not start. Here is an example of the loopback installed in *Debian Sid*;

```
127.0.0.1   localhost
127.0.1.1   mycomputer.mls mycomputer
```

The NSM user-interface (not shown here) that comes up is empty at first. So create a session by clicking the *NSM New* button, and entering a session name (here, "**Seq66**") in the prompt that comes up. In the console window, a couple of `/nsm/server/new` *OSC* messages about the creation of the session appear.

```
[non-session-manager] Sending new for: Seq66
[nsmd] Creating new session "Seq66"
[non-session-manager] /nsm/server/new says Created.
[non-session-manager] /nsm/server/new says Session created
```

Next, click the *Add Client to Session*, and, since `qseq66` has been installed system-wide, it is in the `PATH` and its executable name can be entered simply: "`qseq66`". A number of console messages from *Seq66* appear, plus some messages from *NSM*.

```
[non-session-manager] Sending add for: qseq66
[nsmd] Process has pid: 2797436
[nsmd] Launching qseq66
[nsmd] Got announce from seq66
[nsmd] Client was expected.
[nsmd] Process has pid: 2797436
[nsmd] The client "seq66" at "osc.udp://127.0.0.1:13318/" informs us it's
 ready to receive commands.
```

Important: the *Seq66* user-interface will not show at first. It is hidden so that the screen is not inundated with the windows of all the applications that are (eventually) running under the session. This is especially annoying with tiled window managers. In order to see the *Seq77* user-interface, click on the **GUI** button in the session line shown in the *NSM* window .

Once *Seq66* is running under *NSM*, then click the **Save** button at the top of the *NSM* interface in order to save the session information. This is an *important* step. After *Seq66* exits, one can see what has been created to support the session; the directory that *NSM* creates by default is `/home/user/NSM Sessions`.

```
$ pwd
/home/user/NSM Sessions
$ lstree Seq66
Seq66/
  +-- seq66.nGJDW/
  |   +-- config/
  |   |   +-- qseq66.ctrl
  |   |   +-- qseq66.drums
  |   |   +-- qseq66.mutes
  |   |   +-- qseq66.palette
  |   |   +-- qseq66.playlist
  |   |   +-- qseq66.rc
  |   |   +-- qseq66.usr
  |   +-- midi/
  +-- session.nsm
```

Important: If one is running a recent "New" version of the *Non Session Manager*, then the session data is not stored in `/home/user/NSM Sessions`, but in in `/home/user/.local/share/nsm`, Thus the above would be located differently:

```
$ cd ~/.local/share/nsm    # the value of $XDG_DATA_HOME
Seq66/
  +-- seq66.nGJDW/
  [ the rest is the same as above ]
```

So *NSM* has created a directory with the session name we gave it: `Seq66`.

Side note: the `nsmd` daemon creates a file named after its process ID in `$XDG_RUNTIME_DIR/nsm/d`, e.g. `/run/usr/1000/nsm/d/2170`.

Under that directory is a file, `session.nsm`, which contains information like the following:

```
seq66:qseq66:nGJDW
```

The format of this text is `appname:exename:nXYZT`, where `XYZT` is a 4-letter randomly-generated token generated by *NSM*. Also created is a directory, `seq66.nXYZT`, which is the root of the `Seq66` session.

The rest of the directories, `config` and `midi`, are generated by *Seq66* The `config` directory is used instead of `/home/user/.config/seq66`) and `midi` directory contains new MIDI files, imported MIDI files, or MIDI files from a play-list. The new `config` directory contains versions of the various configuration files that will always be used to start up *Seq66* during the session. One can also add

valid play-list, palette, and drums/note-mapping files to that directory later.

If before running *NSM*, one had set up a play-list file and provided the proper "MIDI base directory" in the 'rc' file, then all the MIDI files are copied to the *NSM* session `midi` directory, preserving all relative directories. When the *Non Session Manager* is started the next time, and the "Seq66" session is clicked, this starts *Seq66*, and the play-list can be seen in the *Playlist* tab.

Note that the **Save** button on the session's row in the *NSM* user-interface sends a message to *Seq66* to tell it to save its state.

One last thing to note is that, when viewing the MIDI ports created by *Seq66*, they will be named "seq66" when not in session management, and "seq66.nXYZT" (for example) when under session management. This makes it possible to run multiple instances of *Seq66*.

9.3.3 Session Management / NSM / Run with Remote NSM

As described in the *NSM* documentation, the `nsmd` daemon can be run stand-alone, and can also be ran on a remote computer. The `qseq66.usr` file can be edited to allow *Seq66* to use a pre-planned *NSM* and specify the URL to connect. Look for the following lines in the 'usr' file:

```
[user-session]
session = none
url = ""
```

Now assume we've run the daemon as follows:

```
$ nsmd --osc-port 9999
[nsmd] Session root is: /home/user/NSM Sessions
NSM_URL=osc.udp://mycomputer.mls:9999/
```

Change the `session` lines to allow the usage of *NSM* at that URL:

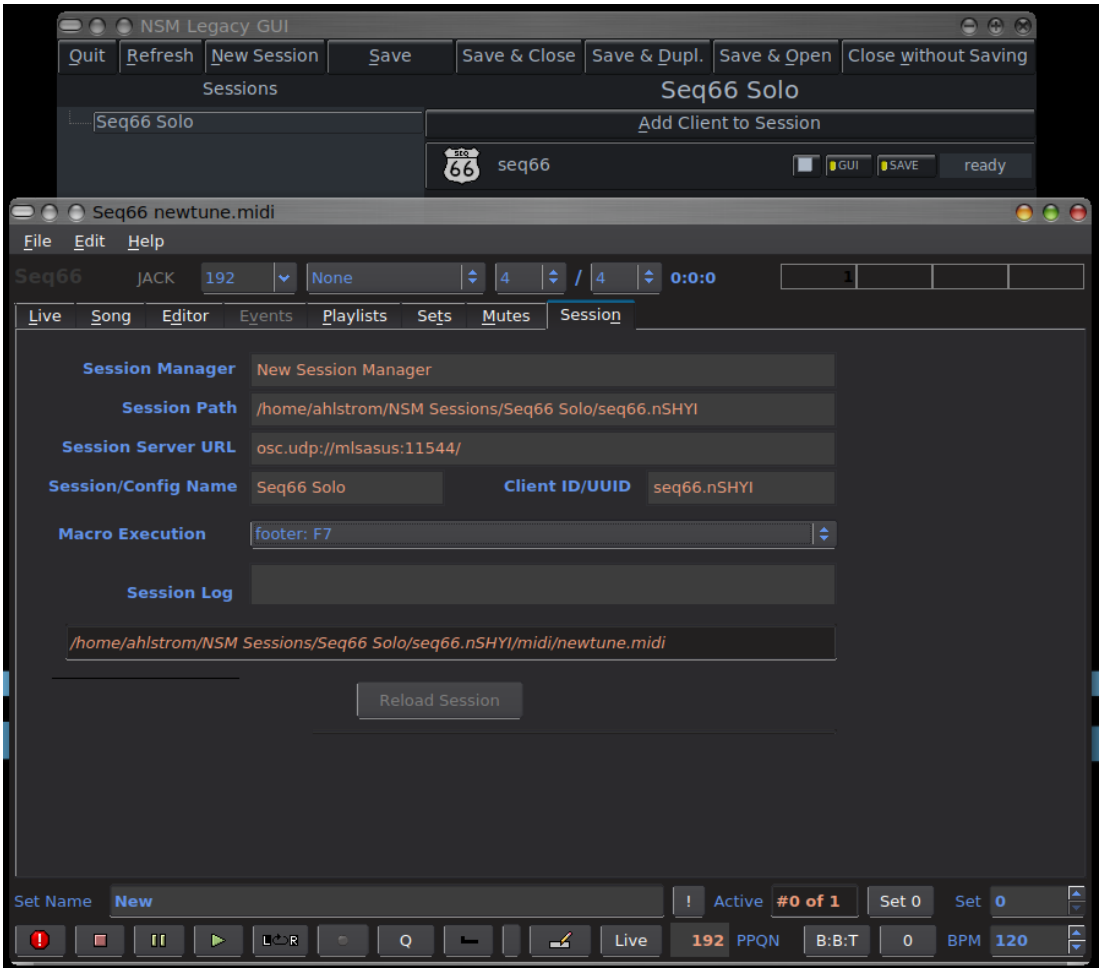
```
[user-session]
session = nsm
url = "osc.udp://mycomputer.mls:9999"
```

The `url` is not used if running *Seq66* from the *NSM* GUI... the application will get the URL from the *NSM* environment. Note that `qseq66` can still be run outside of a session manager. It will detect the absense of the session manager and run normally.

9.3.4 Session Management / Sessions Tab

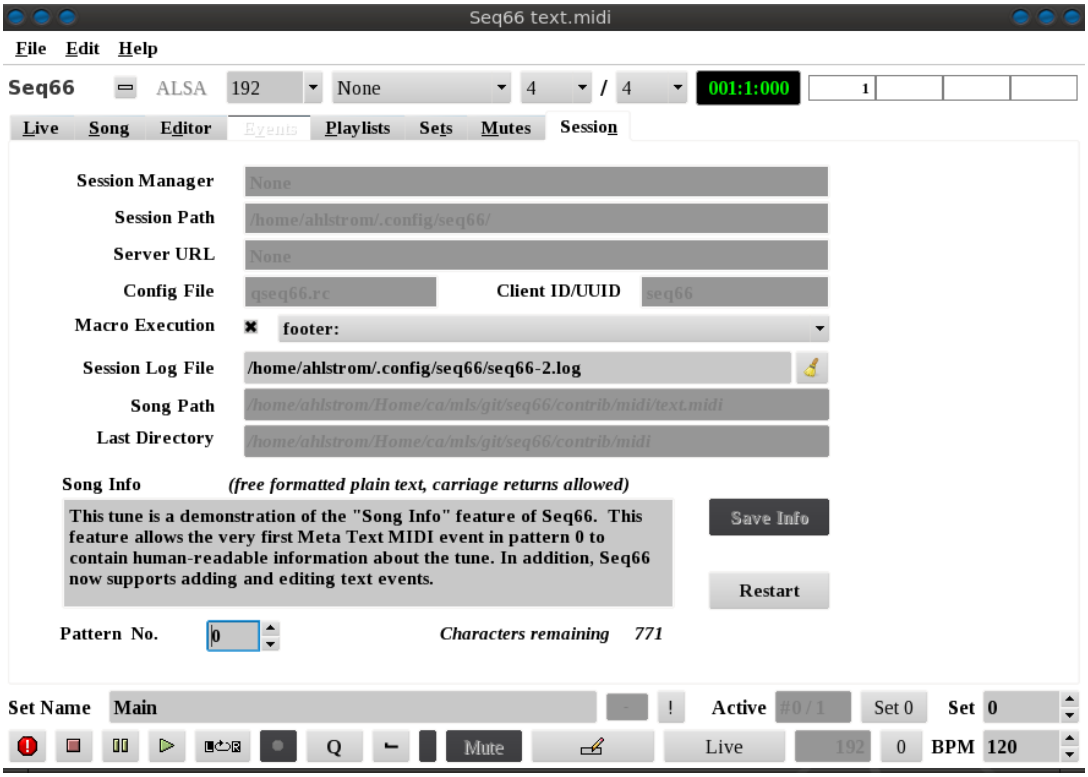
The *Session* tab is a mostly *read-only* tab provided to orient the user to the setup supported by the session. When not running in a session, the normal configuration directory and files are shown. When running in an *NSM* section, the configuration information received from *NSM* is displayed. It is meant to display information to help the user understand what is happening in the run. Two

screenshots are shown below. The first one is a little out-of-date; the second one shows access to widgets to allow song Text information to be stored. (A more flexible text facility is provided in the **Events** tab; see section 8 "Event Editor" on page 108.)



Session Tab Under NSM

Note that the alternate coloring was set using a Qt style-sheet. The next diagram is more up-to-date, and shows the **Song Info** pane for adding a Text event to the song.



Session Tab with Song Info

This section describes the *Session* tab in the main *Seq66* window. This tab displays mostly informative and *read-only* information (except for the name of the log file and the editable song-info pane). It displays the following bits of information that *Seq66* has received from *NSM* via the `nmsd` daemon:

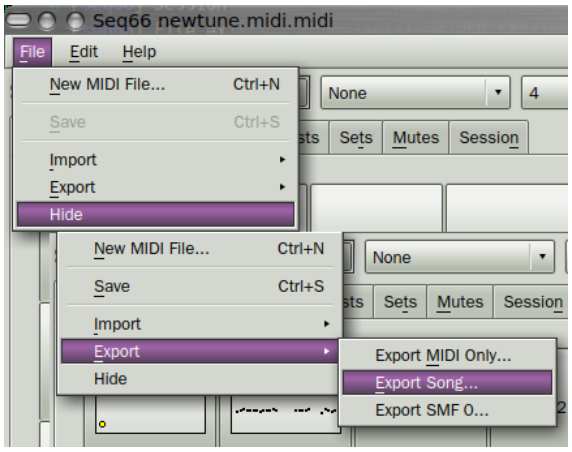
- **Name** of the session manager.
- **Session path** for the session, the root directory of the session. All data goes into this directory. The name of the directory is of the form `HOME/NSMROOT/SESSIONNAME/UNIQUEID`. HOME is the usual UNIX home directory for the user. NSMROOT depends on which version of the New/NSM session manager is used. For the original NSM, this directory is `NSM Sessions`. For the New Session Manager, this directory is `.local/share/nsm`. The session name is provided by the user when creating the session. The unique ID is generated by the NSM. If not running in a session, the active configuration directory is shown.
- The session's **OSC URL**, which includes the port number. Generally, the port number is selected at run-time, but it is also possible to configure *NSM* to use a specific port number.
- **Display name** for the session.
- The generated **client ID** for the session.
- **Macro Execution**. This drop-down contains all of the named MIDI macros defined in the 'ctrl' file's `[macro-control-out]` section. By selecting one, it is automatically sent out via the `[output-buss]` port defined in the 'ctrl' file. The "startup" and "shutdown" macros, if defined, are sent automatically. "Startup" is useful to put a MIDI controller into the proper mode for controlling and displaying information in *Seq66*, and "shutdown" can return the controller to its normal operating mode.

- **Session Log File**.The editable name of the log file to which to redirect warning and error messages during the of action of *qseq66*. Normally, the text is shown in the console window (when running in a console window). This name is only a base-name (e.g. `seq66.log`); it is always stored in HOME.
- **Last Directory**. Shows the directory from where the last MIDI file was loaded.
- **Restart**. After editing some of the preferences in the **Edit** / **Preferences** dialog, one can (later) visit this tab and press this button to essentially restart *Seq66*, reloading the new configuration. Be careful!
- **Song Info** and **Pattern No.**. This item is a plaintext edit-control that allows the viewing and editing of "song info". The song info is merely the first Meta Text event, if any, found in pattern 0. The pattern number can be changed if desired, to show text in other patterns.. This field can be edited with information such as date, composer, playback notes, etc. up to about 32000 ASCII characters. Extended ASCII characters are encoded as three hexadecimal bytes: "
  xx". When the **Save Info** button is clicked, the meta text is copied to the selected pattern, thus modifying the file, which can then be saved. For tracks (such as a karaoke lyric track) with many text events, they are all shown, separated by semi-colons. Don't try to edit such data, except in the *Event Editor*.

Note that there are many implementations of NSM clients: *Agordejo* [3], *RaySession* [31], and the *New/Non Session Manager* [23] with the JACK project's `nsm-legacy-gui`.

9.3.5 Seq66 Session Management / NSM / File Menu

The author of *NSM* has provided documentation for session-management which provides very strict instructions on how an application must behave under session management. *Seq66* tries very hard to stick to these instructions. One major adjustment an application must make is to adhere to the "File menu" guidelines.



File Menu Under NSM, Composite View

Not (yet) shown in the figure are the **Project Configuration** options for import and export. (See section 4.1 "Menu / File" on page 29.)

This has been changed for 0.98.6; the **Quit** menu entry becomes **Hide**, as per the NSM protocol. Also have fixed a bug that disables the load-most-recent option under NSM. We will update the figure above eventually. The following items describe the menu entries.

- **New MIDI File**. This function prompts for the name of a new MIDI file and clears the current MIDI file. The file-name must not include a full-path to the file. The path is hardwired by the session. A relative path can be included. This name is needed because there is no "Save As" option when running in an *NSM* session.
- **Import / Import Project Configuration...** Imports a whole project configuration into the current NSM session. This functionality used to be automatic (importing the "home" configuration), but it is better left to the user to do. However, the restart of *Seq66* after this operation is automatic. Be careful!
- **Import / Import MIDI to Current Set...** This action works the same as in normal mode. This item allows the user to grab a MIDI file from anywhere and import it into the current set. The default directory that comes up in the prompt is the "last-used directory" from the session 'rc' file.
- **Import / Import Playlist...** This action works the same as in normal mode. The destination is the NSM session directory. Once the playlist is imported, *Seq66* is automatically ***restarted*** in order to load the playlist. Be careful!
- **Import / Import into Session...** Prompts the user for a MIDI file to be imported (copied) into the current session. The path to the file is then adjusted to use the *NSM* `midi` subdirectory.
- **Export / Export Song...** Allows exporting the current song as a stock MIDI file, using the performance information (triggers) to write the MIDI data as it would be played in "song" mode. The default directory that comes up in the prompt is the "last-used directory" from the session 'rc' file.
- **Export / Export MIDI Only...** Allows exporting the current song as a stock MIDI file. The "proprietary" SeqSpec data is *not* written. The default directory that comes up in the prompt is the "last-used directory" from the session 'rc' file.
- **Export / Export SMF 0...** This action works the same as in normal mode. It converts the destination file to SMF 0 format.
- **Hide**. This menu item replaces the **Quit** item. It hides the main window and tells NSM about it.

At some point we would like to present a small tutorial showing a session under *JACK*. Also note that NSM can invoke or kill applications via *signals*, as explained in section 9.1 "Session Management / Signals" on page 116.

9.3.6 Seq66 Session Management / NSM / Debugging

This section is oriented towards advanced users who found a problem running *Seq66* and want to track it down themselves. The issue is that we need to start the application under the debugger, or start it under NSM and somehow attach to *Seq66* before it starts running. Another issue is that we have found that, at least on the same host, an NSM session *must* be open before *Seq66* can attach to it, even if the correct `NSM_URL` is provided. So we have to open a session, get the proper URL, configure it in the 'usr' file, and then start *Seq66* under the debugger. Here are the steps:

1. Start *non-session-manager* from a command-line console. Write down the URL that it advertises.
2. Prepare a session for the executable as per earlier instructions. Once `qseq66` starts, immediately exit it, and leave the session open.
3. Open the proper 'usr' file (usually `qseq66.usr`) in a text editor. Set variable "session = nsm", and set the variable "url" to the value that was advertised.
4. Now start `qseq66` in a debugger.
5. Set a breakpoint in `clinsmanager::detect_session()`.

Now you can step through and see where NSM and Seq66 are getting mixed up. Also check the session directory afterward to make the configuration (and any MIDI files) are in good shape.

9.4 Seq66 Session Management / LASH

LASH support has been removed. Use the *NSM Session Manager* or the *JACK Session Manager*.

9.5 Seq66 Session Management / sessions.rc

`Seq66` also supports a more simplistic type of "session", where a whole different set of configuration files can be selected.

One can use the `-home` and `-config` options to specify alternate locations and names for the configuration files. However, if one has a number of configurations (e.g. for different sets of equipment. playlists, style-sheets, and palettes), it's tedious to type these options. The `sessions.rc` file provides a way to set up a number of configurations and select one with one option. It is always located in the default "home" directory, but can refer to directories anywhere. It can also specify a different MIDI client-name (option `-client-name`) and log-file (option `-option log=filename`).

The user must manually text-edit the sessions.rc to specify tag sections like the following:

```
[test]
home = "~/.config/seq66/test/"
config = "test66"
client-name = "test66"
log = "test66.log"
```

This section is accessed using the `-S` or `-session-tag` option, as shown here:

```
$ qseq66 --session-tag test
$ qseq66 -S test
```

If this is the first time this command has been run, the home directory is created. (At present, though, the initial configuration files are not created until *Seq66* exits.) *Seq66* runs with a MIDI client-name of `test66`. The base-name of each configuration file is `test66` (so we have, for example, `test66.rc`). The log file will be `/.config/seq66/test/test66.log`. (If no log file is wanted, then set it to `""` later.)

At the end of the run, all of the configuration files are saved in `/.config/seq66/test/`. These can be edited to suit the "test" configuration. The setup can also be created via **File** / **Export** / **Project configure**, and then be added to `sessions.rc`.

The next time `qseq66 -session-tag test` is run, the test configuration is loaded and used.

If the specified session tag does not exist in `sessions.rc`, then a message is shown; the user should exit **Seq66** immediately and fix `sessions.rc`.

Note that the `log` option overrides the log-file setting present in the 'usr' file. Use `log = ""` to see console output.

Lastly, an environment variable can be used to set a session-tag that will be used for quite awhile. It can be exported in a `bash` script, a profile, or set on the command-line (which is more difficult that using the option):

```
$ SEQ66_SESSION_TAG="test" qseq66
```

10 Import/Export

This section explains the details of the MIDI import and export functionality, accessed by the main menu as noted in sections 4.1.7, 4.1.9, 4.1.10, and 4.1.11, on page 34.

10.1 File / Import Menu

The actions for importing files have been move to the new **File** / **Import** menu in order to keep from cluttering the ever-expanding file menu.

10.1.1 Import MIDI to Current Set

The **File** / **Import** / **Import to Current Set** menu entry imports an SMF 0 or SMF 1 MIDI file as one or more patterns, one pattern per track, and imports them into the currently-active set. Even long tracks, that aren't short loops, are imported. The difference from **File** / **Open** is that the destination screen-set (bank) for the import can be specified, and the existing data in the already-loaded MIDI file is preserved. If the imported file is a *Seq66* MIDI file, it's proprietary sections will *not* be imported, in order to preserve the performance setup. The **Import** dialog is similar to the **Open** dialog.

When imported, each track, whether music or information, is entered into its own loop/pattern box (slot). The import operation can handle reasonably complex files. When the file is imported, the sequence number for each track is adjusted to put the track into the desired screen-set. The import can place the imported data into any of the 32 available screen-sets. Quite large songs can be built by importing patterns.

Import also handles SMF 0 MIDI files. It parcels out the SMF 0 data into sequences/patterns for each of the 16 MIDI channels. It also puts all of the MIDI data into the 17th pattern (pattern 16),

in case it is needed. Note that this slot is used no matter which screen-set one imports the file into. Bug, or feature? Also note that, since the file information has been modified by the import, the user will be prompted to save the file when exiting *Seq66*. Finally, conversion to SMF 1 for SMF 0 files can be disabled using the 'usr' option `[user-midi-settings] convert-to-smf-1`.

10.1.2 Import Project Configuration

This operation allows one to import all of the `qseq66.*` configuration files from a given directory into the current run of *Seq66*.. It is most useful when importing a configuration into a new *NSM* session. Once the files are copied, *Seq66* is automatically restarted, in order to load the new configuration. Be careful!

Note that the names of the configuration files being impported should match the canonical names. That is, the base names should all be `qseq66` (or `qpseq66` for *Windows*.)

10.1.3 Import Playlist

This operation allows one to copy a playlist from one directory to another. It is most useful when importing a configuration into a new *NSM* session. It copies the playlist file (e.g. `liveset.playlist`) into the destination configuration directory. Then it creates a subdirectory with the name `playlist/liveset` (for example). It then copies all of the MIDI files that were referenced in the original playlist into this new directory, preserving the directory structure. It then wires in and activates the new playlist in the 'rc' file, and sets the new base directory for the MIDI files. Lastly, it restarts *Seq66*, to load in the new playlist. Be careful!

10.2 File / Export Menu

The actions for exporting files have been move to the **File** / **Export** menu in order to keep from cluttering the ever-expanding file menu.

An important point to note is the pattern/channel behavior inherited from *Seq24*. When a pattern has a channel specified, all MIDI channel events (e.g. Note On/Off) are stamped with that channel when exported. This applies to exports to plain MIDI files, song exports, and exports to the SMF 0 format.

If one wants to preserve the channel number for all events in a pattern, then one should set the channel to the **Free** value. See section 6.1 "Pattern Editor / First Row" on page 76, the section on **Channel Selection**.

Also note that exporting a song or an SMF 0 file require that the song be exportable (see the next section). Exporting to a regular MIDI does not have this requirement.

10.2.1 Export Project Configuration

This option allows the various `qseq66.*` configuration files to be copied to another directory. This feature is useful for creating an alternate configuration or for backup of the current configuration.

10.2.2 Export Song

Thanks to *Seq32*, exporting song performances (see the **Song Editor**) to standard MIDI format has been added. The **File** / **Export** / **Export Song** operation modifies the song in the following ways:

- Only tracks (sequences, loops, or patterns) that are "exportable" are written. To be exportable, a track must have triggers (see section 24.2.5 "Concepts / Terms / performance, song, trigger" on page 248) present in the **Song Editor**. Also *in the song editor*, it must *not be muted*.
- Each trigger generates the events, including repeats, offset-play of the events, and transposition. If there is a gap in the layout (e.g. due to an **Expand** operation in the **Song Editor**), then the corresponding gap in the events is exported. The result is a track that reconstructs the original playback/performance layout of that pattern. The events themselves are sufficient to play the performance exactly in any MIDI sequencer. The triggers are useful for further editing of the song/performance, so they are preserved in the triggers *SeqSpec* section, but they cover the whole song.
- Empty pattern slots between tracks are removed.
- No matter what set the original track was in, it ends up in the first set; sets are consolidated.
- Other additions, such as time signature and tempo meta events, are written in the same manner as for a normal **File** / **Save** operation.

The **Export** dialog is similar to the **Open** dialog; one will likely want to change the name of the file so as not to overwrite it. If there are no exportable tracks, the following message is shown:

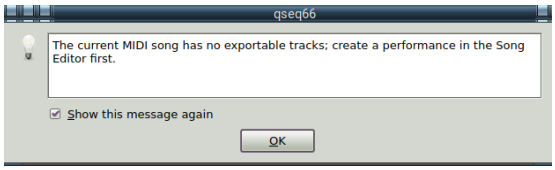


Figure 44: MIDI File Unexportable

Once the file is exported, reopen it to see the results of the export. The following figure shows a before and after picture of the export, as seen in the song editor.

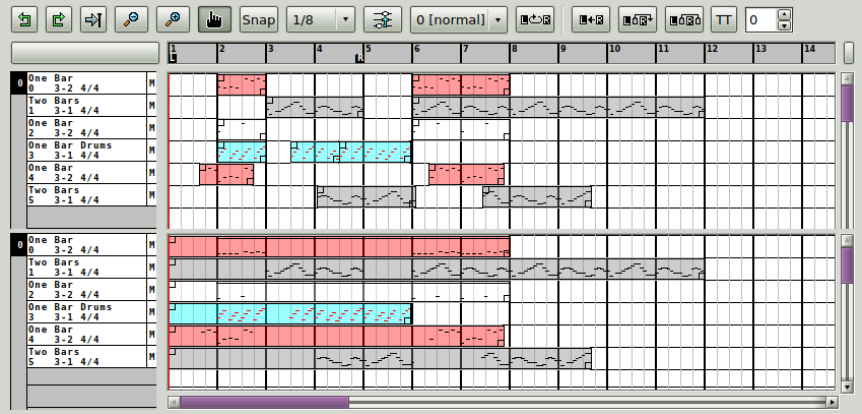


Figure 45: MIDI File Layout Before/After Export

The gaps in layouts in the song/performance data are reflected in the consolidated triggers. Here is the before/after triggers for pattern #0, which was layed out with **Record Snap** *on*:

```
 BEFORE                              AFTER
Sequence #0 'One Bar'               Sequence #0 'One Bar'
Length (ticks): 768                 Length (ticks): 5375
trigger: 768 to 1535 at 768         trigger: 0 to 5375 at 5375
trigger: 3840 to 5375 at 768
```

Note that 768, at PPQN = 192, is 4 beats (1 measure), while 5375 is 28 beats (7 measures). For each of these triggers, the first number is the start of the trigger in PPQNs, the second is the end of the triger, and the third, called the "offset", is actually the length of the pattern. Note how the "AFTER" trigger consolidates the "BEFORE" triggers, starts at time 0, extends to the end of the last trigger, and has a length equal to the end of the trigger.

Now here is the before/after triggers for pattern #5, which was layed out with **Record Snap** *off*:

```
 BEFORE                              AFTER
Sequence #5 'Two Bars'              Sequence #5 'Two Bars'
Length (ticks): 1536                Length (ticks): 6911
trigger: 2344 to 3879 at 1536       trigger: 0 to 6655 at 6911
trigger: 4944 to 6655 at 0
```

6655 is a little over 34.5 beats, which is what the bottom grey trigger shows. 6911 is almost 36 beats (9 measures). Something to figure out.

10.2.3 Export MIDI Only

Sometimes it might be useful to export only the non-sequencer-specific (non-SeqSpec) data from a *Seq66* song. For example, some buggy sequencers (hello *Windows Media Player*) might balk at some SeqSpec item in the song, and refuse to load the MIDI file. For such cases, the **File / Export / Export MIDI Only** menu item writes a file that does not contain the SeqSpec data for each

track, and does not include all the SeqSpec data (such as mute groups) that is normally written to the end of the *Seq66* MIDI file.

10.2.4 Export SMF 0

In some cases it might be useful to convert a *Seq66* MIDI file to a single-track SMF 0 file. As with exporting to a song (see section 10.2.2 "Export Song" on page 130), the tracks to be exported (combined into a single track) must be unmuted and have layouts in the song editor. The **File** / **Export** / **Export SMF 0** menu action removes all the existing tracks and merges them into track 0.

11 Seq66 Recording In Depth

Recording in *Seq66* has been greatly enhanced. Multiple patterns can be recorded at once, with events routed to particular patterns based on the buss number the event came in on, or its channel number. Additional ways to toggle recording have been added. Additional recording alterations have been added, such as on-the-fly note mapping. It can get a bit complex to keep track of, hence this section, which walks the user through some scenarios.

Before getting started, note that recording also can be done during count-in via the metronome feature. See section 4.2.1.8 "Menu / Edit / Preferences / Metronome Options" on page 50.

11.1 Recording Scenarios

The following recording scenarios are available:

- **Standard Recording**. This provides the normal method of recording. Open a pattern in a window and enable recording. The last slot selected for recording receives the events. Events are received from all enabled MIDI input ports. In this mode the record-button at the bottom of the main window is red. The slot popup-menu does not provide for setting an input buss.
- **Route-by-Bus**. In this mode, any pattern that specifies a specific input buss can be enabled for recording. Incoming events are routed to the first pattern that has an input buss matching the buss on which the event was recorded. In this mode the record-button at the bottom of the main window is green.
- **Record-by-Channel**. In this mode, one enables recording in patterns numbered from 0 to 15 (channels 1 to 16) with output channels set from 0 to 15. Incoming events are analyze for their channel and are recording into the corresponding pattern. In this mode the record-button at the bottom of the main window is yellow.

11.1.1 Standard Recording

Standard recording allows the enabling of recording in a single pattern. It *requires* that the other two modes be turned off. Once a pattern is set to record, no other pattern can be set to record until

the original pattern is set to not record. The single recording pattern gets all events from any MIDI input that is enabled.

In standard recording there a many ways to enable recording into a pattern.

- Open the pattern in a window or tab, and click the record button at the bottom of the pattern editor. It will turn red when enabled. The pattern will also show a red circle to indicate that recording has been enabled.
- Right-click on the desired pattern in the grid and select **Record toggle** from the menu.
- Touch a pattern or click its hot key, then click the red record button at the bottom of the main window.
- Select the **RECORD** grid mode, then select a pattern. (This option is more useful in the other recording scenarios.)

Again, remember that, with standard recording, only one pattern can accept events. Also note that this mode should be used if one expects to record SysEx events, which have no channel.

11.1.2 Route-By-Buss

Route-by-buss is enabled whenever a pattern in a song specifies an input buss *and* **Edit** / **Preferences** / **MIDI Input** / **Record into patterns by bus** is checked. It is an 'rc' file option. If it is enabled, it supercedes the **record-by-channel** option and disables that standard recording mode described above.

When route-by-bus is enabled, an internal container is populated with all the patterns in the current play-set that specify an input buss. This container is rebuilt when a sequence is added or removed. It is stored in the `c_midiinbus` SeqSpec in the song.

If route-by-bus is enabled, a sequence with an input bus that matches the buss associated with the incoming event is looked up, and input is streamed to it.

In this scenario, multiple patterns can be enabled for recording, and output from multiple input ports are routed appropriately.

Clicking on the green record button at the bottom of the main window will turn on recording for *all* patterns that specify an input buss.

11.1.3 Record-By-Channel

Record-by-channel can be enabled in **Edit** / **Preferences** / **MIDI Input** / **Record into patterns by channel**, which is an 'rc' file option.

It works by routing events to the first pattern that has specified an output (not input) channel that matches the channel (if applicable) of the incoming MIDI event. The patterns applicable are entered into a list of patterns with specified output channels.

For convenience, there is a MIDI file installed, called **16-blank-patterns**, which specifies all the

output channels. It can be copied and used for the first recording from a device playing on multiple channels, or from multiple devices, each playing on a different channel.

Clicking on the yellow record button at the bottom of the main window will turn on recording for *all* patterns that specify an input buss.

Note that the route-by-buss option, described above, supercedes this option.

11.2 Recording Modes

Recording can also transform (alter) the incoming events.

- **Tighten**. Partial quantization to the current snap value.
- **Quantize**. Full quantization to the current snap value.
- **Note-map**. If a 'drums' file is active, then notes are changed in note-value according to that file. This is most useful for drums, but other effects are possible.

These transformations can also be done after recording.

12 Seq66 Configuration

Seq66 configuration has become more elaborate with time, with more configuration files. Configuration items are well documented in the *Seq66* "man" page and in the configuration files. Therefore, this new discussion will merely summarize the options and go into a few details about the configuration files. Here are the topics to discuss:

- **Command-line Options**. Useful with desktop shortcuts.
- **'rc' File**. Mostly I/O port options, JACK options, recent files. Also now specifies other files to be used.
- **'usr' File**. Local names for busses and instruments, user-interface settings, sessions... Some options that fit either the GUI or the command-line application could be moved to the 'rc' file.
- **'ctrl' File**. The keystroke and MIDI control settings have been moved to a separate file for flexibility.
- **'mutes' File** The mute-groups settings have been moved to a separate file for flexibility.
- **'drums' File**. This file supports remapping percussion notes from older drum machines to General MIDI.
- **'playlist' File**. This new file specifies a file that contains one or more playlists and MIDI controls for them.
- **'palette' File**. This new file allows replacing the default piano-roll, time, data, and events drawing to be tailored separately from the Qt theme.
- **'qss' File**. Qt style-sheets can now tailor the appearance of Qt theme-drawn elements (e.g. the pattern-grid buttons and text controls).

After the first run and exit of *Seq66*, it generates a set of default configration files in the default *configuration* directory:

```
/home/user/.config/seq66/qseq66.rc
/home/user/.config/seq66/qseq66.usr
/home/user/.config/seq66/qseq66.ctrl
/home/user/.config/seq66/qseq66.mutes
/home/user/.config/seq66/qseq66.drums
/home/user/.config/seq66/qseq66.playlist
/home/user/.config/seq66/qseq66.palette
```

The palette file is not automatically generated. It can be saved from the **Edit** / **Preferences** / **Display** tab. The style-sheet file can be specified in the 'rc' file's `[style-sheet-file]` section. There are also some 'keymap' files. They are not yet used, but may become a feature of *Seq66* in the future. For *Microsoft Windows*, the default base name of the files is `qpseq66`, and the default configuration directory is

```
C:/Users/user/AppData/Local/seq66/
```

When running *Seq66* from the *Non Session Manager* (Linux-only, see section 9 "Session Management" on page 116), the configuration directory is automatically set to something like

```
/home/user/NSM Sessions/MySession/seq66.nRSIQ/config
```

There is no palette file by default, but the user can create one. The color palettes are discussed in section 16 "Palettes for Coloring" on page 199. Other files contain the the data for remote MIDI control, computer keyboard control, MIDI clock, JACK transport, and a many other settings. Some of the settings can be modified in the **Edit** / **Preferences** dialog, or overridden from the command line.

Seq66 overwrites the most of these files upon exiting. One must therefore quit *Seq66* before making manual modifications to these files.

12.1 Configuration File Commonalities

All of the *Seq66* configuration files have the following in common:

- **[Seq66] Section**
- **[comments] Section**
- **Numeric Settings**
- **Boolean Settings**
- **Variables**
- **Stanzas**

Generally, each configuration file has its own specific set of sections, each section-name being enclosed

in square brackets in a very strict format: No spaces inside the square brackets. Sections are looked up by this name, including the square brackets, and the name must be exact.

12.1.1 [Seq66] Section

This section is informational. At a minimum, it holds two variables:

- `config-type`. This value indicates the type of the file, such as "ctrl" or 'rc'.
- `version`. This value indicates the version of the file. It is used in some cases when we have added a feature to the configuration file that cannot be automatically handled. When the internal version number of the file is incremented, it can be used to make adjustments for changes in configuration when reading older versions of files.

This section may also contain additional "global" variables specific to a given `config-type`.

12.1.2 [comments] Section

The "comments" section is also informational, but the user can edit this section to include information describing the purpose of the file. For example, a 'ctrl' file for a *Novation Launchpad* might describe the purpose of this file. The comments stop at the first blank (not even spaces) line. To skip a line in the comment, put a single space character on the blank line.

12.1.3 Numeric Settings

Numeric settings consist of a line containing one or more numbers, usually preceded by an explanatory comment, and followed by a standard script comment.

```
3    # grid_style       # obsolete, by the way
```

We have been replacing these kinds of settings with the `name = value` style described below, but there are a few places where the old style persists, or is more convenient.

12.1.4 Boolean Settings

Boolean settings are the same as numerical settings, but have only two values: "0" or "1".

```
0        # flag to record incoming data by channel
```

We have been trying to replace these kinds of settings with the `name = true/false` style described below, but there are a few places where the old style persists.

12.1.5 Variables

Variable are a new style of value setting, and can encompass not only booleans and numeric values, but string values, which may correspond to enumerated values in the source code. These values are specified by a section-name plus variable name/value pair. Here are some sample `name = value` pairs:

```
load-mute-groups = true
save-mutes-to = both
window-scale = 1.0
default-ppqn = 1920
directory = "/home/user/Seq66 files/midi"
```

If the variable value is a string value that is the name of a file or directory or path, it must be surrounded with quotes, in case the path has spaces in it.

12.1.6 Stanzas

We have streamlined the control-file stanza by eliminating the "enabled" and "channel" columns, since they can be encoded in the event/status byte (e.g. 0x90) instead. Older versions of the 'ctrl' file will be upgraded automatically.

A stanza in a *Seq66* configuration file consists of some data at the beginning, a set of values inside square brackets, and a comment at the end. The values inside the square brackets are numeric, and can be in decimal format, hexadecimal format, or binary "0/1" format. See section 12.5.2 "'ctrl' File / Loop Control" on page 164, which describes the details of this layout.

```
0 "1" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ]  # Loop 0
1 "q" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ]  # Loop 1
2 "a" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ]  # Loop 2
3 "z" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ]  # Loop 3
. . .
```

12.2 Command Line

Command-line options are well-described in the *Seq66* "man" page. Here, we will present a brief note about each option, and, where applicable, a reference to the corresponding configuration file option. Here is the basic command line:

```
qseq66 [options list] [MIDI filename]
```

`-h --help` Display a list of all command-line options, then exit.

`-V --version` Display the program version, then exit.

`-v --verbose` During execution, write more information to the console. Useful for trouble-shooting.

`-n --nsm` Activate a simple simulation of the built-in NSM (*Non Session Manager*) support, for debugging. As of version 0.99.3, *Seq66* detects if its parent process is the `nsmd` daemon automatically (on Linux).

`-T --no-nsm` Ignore the NSM setting in the 'usr' file. ('T' for 'typical'). Not too useful at this time.

`-X --playlist [filename]` This option loads the given file-name as a play-list file. See section 13 "Seq66 Play-Lists" on page 184.

qseq66.rc: `[playlist] name`

qseq66.playlist: `[playlist] full`

`-m --manual-ports` *Seq66* won't attach the system's existing ALSA or JACK MIDI ports. Instead, it will create is own set of *virtual* input and output busses/ports. The default number of port is 1 for input, and 16 for output, but these values can be changed in the 'rc' file.

qseq66.rc: `[manual-ports] flag for manual/virtual ports`

`-a --auto-ports` *Seq66* will automatically attach to the system's existing ALSA/JACK ports.

qseq66.rc: `[manual-ports] flag for manual/virtual ports`

`-r --reveal-ports` *Seq66* will show the names of the ALSA/JACK ports that the system defines, rather than the names defined in the 'usr' configuration file.

`-R --hide-ports` *Seq66* will show the names of the ALSA port that the 'user' configuration file define, rather than the names defined by ALSA.

qseq66.rc: `[reveal-ports] flag for reveal ports`

qseq66.usr: `[user-midi-bus-definitions] number of user-defined busses`

`-A --alsa` *Seq66* will run with ALSA, even if JACK is running. This options is "sticky" (they are saved).

`-b --bus [buss]` Modifies the output buss number on *all* tracks when a MIDI file is read. Useful for testing or quick-and-dirty setup, and for adapting a playlist to route to a specific MIDI output.

`-l --client-name` Replaces the normal client-name, `seq66`, with the given name. Probably best to make sure "seq66" is part of the name, for clarity. Also note that, if active, the *Non Session Manager* will override this name, e.g. using something like *seq66.nXYZT*.

`-q --ppqn [ppqn]` Supports modifying the PPQN value of Seq66, which is defaults to a value of 192. This setting is written into the MIDI file when it is saved. The PPQN value can range from 32 to 19200, or be set to 0 to use the PPQN from the loaded file.

qseq66.usr: `[user-midi-settings] midi_ppqn` .

qseq66.usr: `[user-session] session`

`-s --show-midi` Dumps incoming MIDI to the screen.

`-p --priority` Runs at higher priority with a FIFO scheduler.

`-k --show-keys` Prints pressed key value.

`-K --inverse` Changes the color palette for the sequence editor and performance editor piano rolls. Also note that the palette is highly configurable. And there is also an option to use a Qt style-sheet.

qseq66.palette: `[palette] inverse`

The following options can be configured in the 'rc' file. (But note that, since version *Seq66* 0.94.0, the format of that section of the file has been simplified. See section 12.3.14 "'rc' File / JACK Transport" on page 150.

`-t --jack-midi` *Seq66* will run with JACK, for MIDI if JACK is running.

qseq66.rc: `[jack-transport] jack-midi = true`

`-N --no-jack-midi` *Seq66* will not run with JACK for MIDI, even if JACK is running.

qseq66.rc: `[jack-transport] jack-midi = false`

`-W --jack-connect` *Seq66* will connect automatically to JACK ports discovered on the system. This is the long-standing default with Seq66.

qseq66.rc: `[jack-transport] jack-connect = true`

`-w --no-jack-connect` *Seq66* will not automatically connect to other JACK ports. This option is useful if a session manager is in charge of the connections. This option is not available with the other MIDI engines.

qseq66.rc: `[jack-transport] jack-connect = false`

`-S --jack-slave`

`-S --jack-slave` *Seq66* will sync to JACK transport as a "slave", if *JACK* is running.

`-j --jack-transport` This option is the old, **deprecated** version of `--jack-slave`.

qseq66.rc: `[jack-transport] transport-type = slave`

`-g --no-jack-transport` *Seq66* will not sync to JACK transport. This disables JACK transport if it had been enabled previously.

qseq66.rc: `[jack-transport] transport-type = none`

`-J --jack-master` *Seq66* will try to be JACK master.

qseq66.rc: `[jack-transport] transport-type = master`

`-C --jack-master-cond` JACK master will fail if there is already a master.

qseq66.rc: `[jack-transport] transport-type = conditional`

`-M --jack-start-mode [x]` When *Seq66* is synced to JACK, the following play modes are available: "live" = live mode; "song" = song mode, and "auto" means use song mode if triggers are present

in the loaded MIDI file. "auto" is the default.

qseq66.rc: `[jack-transport] song-start-mode = auto`

`-U --jack-session-uuid [uuid]` Set the UUID for the JACK session. This value is useful when running *Seq66* from within a *JACK* session. The session controller (e.g. *QjackCtl*) uses this value in the command-line when starting *Seq66*. See section 9.2 "Session Management / JACK Session" on page 117.

`-0 --smf-0` Normally, SMF 0 (single-track MIDI) files are split into separate tracks when read into Seq66. This 'usr' option preserve the file as an SMF 0 file.

`-u --user-save` Save the 'usr' configuration file when exiting. Normally, it is saved only if not present in the configuration directory, so as not to get stuck with temporary settings such as the --bus option.

qseq66.rc: `[auto-option-save] auto-save-rc`

`-H --home [directory]` Change the "home" configuration directory from `$HOME.config/seq66` The configuration files are loaded from or saved to the specified directory. If the directory is a full path (such as `~/test/import`), then the directory is used as is. If the directory is a simple directory (such as `test` or `test/import`), then it is placed in the normal "home" configuration directory and is used to contain the configuration.

`-c --config basename` Use a different configuration file base name for the 'rc' and 'usr' files. For example, one can specify a full configuration for "testing", for "jack", or for "alsa", to set up `testing.rc` and `testing.usr`, `jack.rc` and `jack.usr`, `alsa.rc` and `alsa.usr`. But note that recent versions of *Seq66* use the 'rc' file to control the names and active state of all the other configuration files.

`-f --rc filename` Use a different 'rc' configuration file. It must be a file in the user's `$HOME/.config/seq66` directory or the directory specified by the `--home` option. The `.rc` extension is added if necessary.

`-F --usr filename` Use a different 'usr' configuration file. Similar to the `--rc` option. But note that this can be controlled in the 'rc' file as well.

`-S --session-tag section` Use one of the sections present in `$HOME/.config/seq66/sessions.rc` to set up an alternate configuration. For details, see section 9.5 "Seq66 Session Management / sessions.rc" on page 127.

`-L --locale localename` Set a different locale for running *Seq66*. If the current locale uses the comma as a decimal point, then try to use `en_US.UTF-8` (for example).

`-o --option opvalue` Provides additional options, since the application is running out of single-character options. The `opvalue` set supported is:

- `daemonize` and `no-daemonize`. Sets up the `seq66cli` application to fork to the background the *next time* it is run. This is done by modifying the `seq66cli.usr` file to specify that `seq66cli` runs as a daemon. It is good to specify a log-file as well. undoes the setup of `seq66cli` daemonization. The application can then be run in a console so that log output can be seen.
- `log=filename.log`. Reroutes standard error and standard output messages to a log-file located in the configuration directory. If this file is present, log information is appended. The default

log-file name is specified in the `[user-options]` section of the 'usr' file. If using daemonization, it pays to also set up a log file with the same command.

- `sets=8x8`. This option, informally known as "variset", allow some changes in the set size and layout from the default 4x8 = 32 sets layout. Consider this option to be experimental. Expect problems (and please report them). To save these options to the 'usr' file, add the `--user-save` option to the command line, or edit the 'usr' file before running *Seq66*. In that file, the options modified are `mainwnd_rows` and `mainwnd_cols`.
- `scale=WxH`. This option scales the main window by the given factors, ranging from 0.5 to 3.0. A 1920x1080 screen can be completely filled via `--option scale=2.175x1.75`. Even when scaled down, the user can use the mouse of window-manager keystrokes to shrink the window even further. Note that the other tabs in the main window do not adjust appropriately... this feature is meant for live usage of a touch-screen.
- `mutes=value`. Specifies the saving of mute-groups to the 'mutes' file, 'midi' file, or 'both' files.
- `virtual=o,i`. Set up the manual-ports option with 'o' output ports and 'i' input ports.

12.3 'rc' File

```
/home/user/.config/seq66/qseq66.rc
```

The 'rc' configuration file has undergone a lot of changes, including off-loading the keyboard control, MIDI control, and mutes control sections into their own files, and adding a few "variable" settings. Rather than repeating information already present in the self-documenting 'rc' file, we will summarize the settings and refer the reader to the sample files for more information.

The 'rc' file adds these `[Seq66]` options to the common data for all configuration files:

```
sets-mode = normal
port-namng = short
```

The `sets-mode` option determines how patterns are muted when the play-screen (current set) changes. Its values are:

- **normal**. When moving to another set, the patterns in the current play-screen are muted, as are the patterns in the destination set.
- **autoarm**. Similar to normal, but the patterns in the destination set are automatically unmuted.
- **additive**. When moving to another set, the destination set is added to the play-set, so that both sets are playing. This option was requested by users.
- **allsets**. When a song is loaded, all patterns in all sets are unmuted and will play.

The `port-naming` option determines how much detail is shown in port names as shown in the user-interface. It does not apply if port-mapping (section 21 "Port Mapping" on page 227) is active (and note that port-mapping is now the default). Port-naming values are:

- **short**. The brief name is shown; this is just the bare device name.

- **pair**. The ALSA client number and port are shown along with the device name. Example "128:0 VMPK Input"
- **long**. The internal number of the clients and ports are shown. These are a sequence of port numbers: 0, 1, 2, 3, etc.

For *Windows* (the "portmidi" build), the 'long' option works better.

12.3.1 'rc' File / MIDI Control

Seq66 offloads MIDI control to a separate file. Move or create the `[midi-control]` section to a separate file in the *Seq66* configuration directory, and add the following snippet:

```
[midi-control-file]
active = true           # true means to use and to rewrite at exit
name = "qseq66.ctrl"    # contains a [midi-control] section, and more
```

As with the 'rc' file, the 'ctrl' file can be rewritten upon exit, *except* that it is not written unless it is *active*, or *does not exist* (as during the first run of *Seq66*). For the details of the 'ctrl' file, see section 12.5 "'ctrl' File" on page 163.

12.3.2 'rc' File / Mute Groups

The 'rc' file has been modified so that mute-group are now stored in a separate file. The following section specifies that file, which should be located in the session/configuration directory.

```
[midi-group-file]
active = false
name = "qseq66.mutes"   # contains a [mute-groups] section
```

The mutes-groups themselves are loaded from the 'mutes' file dependent on the `load-mute-groups` option in the mute-groups file. The mutes-groups section is written on exit dependent on the `save-mutes-to` option in the mute-groups file. It can go to the MIDI file, the 'mutes' file, or both.

12.3.3 'rc' File / Color Palette

```
[palette-file]
active = false
name = "qseq66.palette"
```

The only need for a palette file is when not satisfied with the default palette for the patterns, inverse colors, hatching, or grid-lines for some pattern piano roll items. There is a button to save the current/default palette for later modification in the **Edit** / **Preferences** / **Display** tab.

12.3.4 'rc' File / Style Sheet

The `[style-sheet-file]` option, if active and non-empty, causes a user-designed style-sheet to be applied. This is useful in expanding the tab-sizes, making disabled text easier to read in some Qt themes, or fixing some minor faults in the current Qt theme. See `data/samples/textfix.qss` for an example. The style sheet file is assumed to reside in the normal *Seq66* configuration directory. Here is an example using the style sheet file `data/samples/qseq66.qss`:

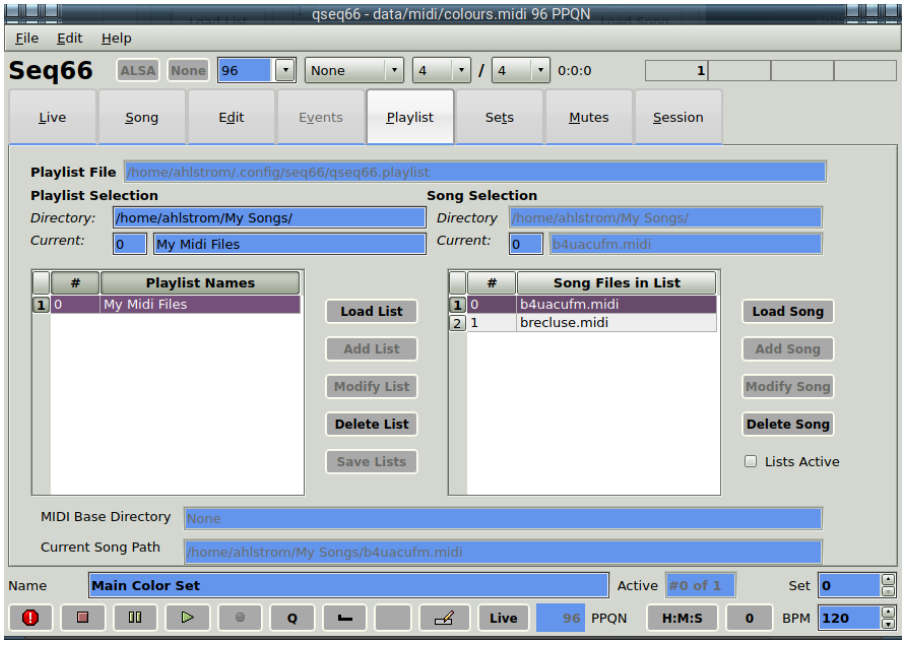


Figure 46: Seq66 View with a Minimal Style-Sheet Applied

Note the blue text fields. Also note the larger tabs, which could be useful on a touch-screen. There is a full-scale dark theme stored in `data/win/dark-theme.qss`. One might want to save and tweak the 'palette' file to match.

The following is a more comprehensive example using the `incrypt-66.qss` style-sheet to control the appearance of *Qt*-drawn items, plus the `incrypt-66.palette` file to alter the colors of the drawn elements (grids and live/song backgrounds) that are not subject to style-sheet alterations.

Figure 47: Enhanced "Incrypt" Style-Sheet

The following shows the **Song** tab and a sample **Preferences** tab using the same style-sheet and palette.

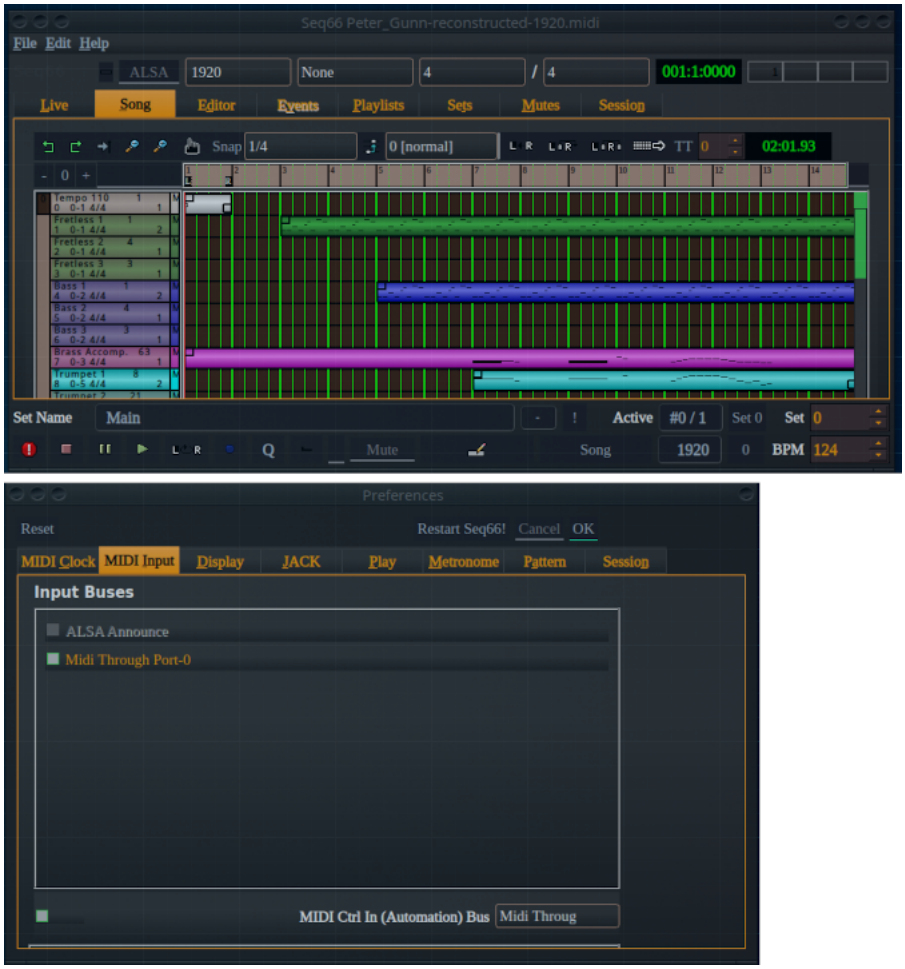


Figure 48: Enhanced "Incrypt" Style-Sheet Part 2

The thing to note here is that we must limit the width of combo-boxes so that all elements in a horizontal bar can be seen. This has the unfortunate side-effect of showing only part of long port names (e.g. "Midi Throug" in the figure above).

Another interesting style-sheet in `data/samples` is `perstfix-66.qss` and its palette file, which is a nice blue theme. No need to show it here; try it.

12.3.5 'rc' File / Note Mapper

```
[note-mapper]
active = false
name = "GM_DD-11.drums"
```

This file can be used transform the existing drum (non-transposable) tracks into another set of drum tracks. A lot of work has been done in the past with non-General-MIDI instruments (particularly consumer instruments like the *Yamaha PSS-780* or *Yamaha DD-11*. This option is useful for transformation older MIDI files into GM format. For the usage of the note-mapper, see Figure 38

"One-Shot Pattern Recording" on page 98, and the surrounding discussion.

12.3.6 'rc' File / MIDI-Clock Section

The MIDI Clock tab contains the clocking state from the last time *Seq66* was run, their status, and their names. It also specifies the MIDI output ports, which can be disabled. Disable the port with a -1, turn off the clock with a 0, or turn it on with a 1 (which means to send **MIDI Song Position**, and **MIDI Continue** if starting after tick 0), or on with positioning with a 2, which sends **MIDI Start** and then begins clocking after the position reaches a modulo of the **Clock Start Modulo value**. Luckily, the user-interface makes it easy to select the desire value, and has tool-tips to instruct the user. This configuration item is represented in the **Edit / Preferences / MIDI Clock** tab.

```
[midi-clock]
5    # number of MIDI clocks (output busses)
0 0    "[0] 14:0 Midi Through Port-0"
1 0    "[1] 128:0 TiMidity port 0"
2 0    "[2] 128:1 TiMidity port 1"
. . .
```

If there are USB MIDI devices plugged in (on *Linux* using JACK), they will show up with system names that have no meaning to the user. In this case, an "alias" looked up by JACK is appended as a comment in the 'rc' files, and also shows up in the port lists in the user interface. For example:

```
0 0    "[0] 0:0 system:midi_capture_1"        # Midi-Through
1 1    "[1] 0:1 seq66:system midi_capture_5"  # Launchpad-Mini
2 0    "[2] 0:2 system:midi_capture_3"        # USB-Midi
3 0    "[3] 0:3 system:midi_capture_4"        # nanoKEY2
```

Note the difference in naming of the enabled port (Launchpad-Mini). The appended label is *not* part of the port name; it is there simply to help the user select the correct system port. This same setup applies to MIDI input ports as well. Note that the 'rc' file has a port-mapping option, described elsewhere.

Ports created by `a2jmidid -export-hw` do not have JACK aliases. Ports created by Seq66 do not have JACK aliases. Ports created by MIDI applications such as *Qsynth* do not have JACK aliases. Ports created by ALSA or *Windows* do not have aliases.

On some recent *Linux* systems, these USB system ports can be accessed without exposing the ports via `a2jmidid`, since JACK does it. One can see all the aliases on a JACK setup with the following command:

```
$ jack_lsp --aliases
```

For output port-mapping, which is now the default for *Seq66*, see section 21.2 "Output Port Mapping" on page 230.

12.3.7 'rc' File / MIDI Clock Mod Ticks

This configuration item is the same as the **Edit / Preferences / MIDI Clock / Clock Start Modulo** option.

```
[midi-clock-mod-ticks]
ticks = 64
record-by-buss = false
record-by-channel = false
```

The record-by-buss flag, if true, causes recording to route events to the first pattern that specifies an input buss number matching the buss number of the event. The record-by-channel flag, if true, causes recording to route events to the first recording pattern that has an output channel matching the event's MIDI channel. To use these features, the user must provide an existing pattern, set its input buss or its output channel as desired, and enable recording for that pattern. To facilitate this operation, a MIDI file `data/midi/16-blank-patterns.midi` is provided, with output channels already set. Make a copy of it and start with that copy.

Be aware that events will go only into patterns for which recording has been enabled. Also be aware that record-by-buss and record-by-channel are mutually exclusive. See section 11 "Seq66 Recording In Depth" on page 132, which details how the record-by features work.

The record-by flags should probably be in the 'usr' file.

12.3.8 'rc' File / MIDI File Tweaks

This section provides some modifications to how MIDI events are handled, specifically, at present, in regard to the failures of handling running status.

```
[midi-file-tweaks]
running-status-action = recover
```

The options available are:

- **recover**. Tries to recover the running status when a data byte is encountered. The previous running status byte is saved, and then used when a data byte is encountered. This option allows the MIDI file parsing for the track to continue, and then the rest of the tracks are processed.
- **skip** ignores the rest of the bytes in the track, and allows the rest of the tracks to be processed. The main difference between this option and **recover** is that the former allows *Seq66 SeqSpec* events to be created.
- **proceed** keeps going in spite of errors, reporting them. Try it with `contrib/midi/trilogy.mid` to see what messages appear in the console and what patterns actually get loaded.
- **abort** just exits the parsing, which is the old and undesirable behavior. It causes patterns to be dropped.

Try each option with the `trilogy.mid` file, in which running status is started, only to be ended by the 0xF0 SysEx event. But then the events after the SysEx event still require running-status to be in effect.

This section is not yet accessible from an **Edit** / **Preferences** tab.

12.3.9 'rc' File / MIDI-Meta-Events Section

The MIDI Meta events section is the start of additional options supporting meta events as normal events in *Seq66*; it defines only the tempo MIDI meta-event at present. Normally, tempo events are supposed to occur in the first track (pattern 0). But one can move this track elsewhere to accomodate one's existing body of tunes. It affects where tempo events are recorded. The default value is 0, the maximum is 1023. A pattern must exist at this number for it to work.

```
[midi-meta-events]
tempo-track = 0
```

As per the MIDI specification, the first track (track 1 in track numbering, or pattern 0 in *Seq66* numbering) is *the* official track for certain MIDI meta events, such as **Set Tempo** and **Time Signature**. But we allow the user to and change the tempo track to another pattern.

12.3.10 'rc' File / MIDI Input

This configuration item is represented in the **MIDI Input** tab in the **Edit** / **Preferences**. The first number is a line count, and equals the number of supported input ports. After that, this 'rc' entry here has two variables; the first is the port number, and the second number indicates whether it is disabled (0), or enabled (1). The next lines show the input busses present on the system (normally).

```
[midi-input]
2   # number of MIDI busses
0 1    "[0] 0:1 system:announce"
1 0    "[1] 14:0 Midi Through Port-0"
2 0    "[2] 28:0 nanoKEY2 MIDI 1"
3 0    "[3] 48:0 Launchpad Mini MIDI 1"
```

Note that the "system:announce" buss is present only for ALSA, not for JACK. This will change the port numbering. Again, see the 'rc' file itself for more information.

One minor issue with the system MIDI Thru port is that, if it and another input port (e.g. VMPK) are both enabled, then each note emitted by VMPK is doubled. Be aware.

For input port-mapping, which is now the default for *Seq66*, see section 21.1 "Input Port Mapping" on page 229.

12.3.11 'rc' File / Manual (Virtual) Ports

The name of this setting is a bit of a misnomer in a couple of ways:

1. It refers to the usage of *virtual* MIDI ports. These are ports set up by the application so that other devices, applications, or session managers can connect *manually* to the MIDI application.
2. This option is not just for ALSA. It can also be used when running in native JACK mode, to support virtual JACK ports that can be connected manually (e.g. in the *QJackCtl* application.)

```
[manual-ports]
virtual-ports = false   # 'true' = manual (virtual) ALSA or JACK ports
output-port-count = 8   # number of manual/virtual output ports
input-port-count = 4    # number of manual/virtual input ports
```

The opposite of `--manual-ports` is `--auto-ports`, which is the normal mode of running *Seq66*. In this mode, system MIDI input/output devices are discovered and automatically connected. It will create port names as per the settings in the 'usr' configuration file's sections:

```
[user-midi-bus-definitions]
[user-midi-bus-N]
```

These definitions can be used by JACK for connection, and these definitions can be used to specifically rename the ports that exist in the system. (Roughly similar to the MIDInam feature, but not compatible.) This option is misleading if one wants to have access to the actual ALSA/JACK ports that exist on the system. The next option gets around that issue.

12.3.12 'rc' File / Port Map

The 'rc' file also supports a port map, which maps I/O ports from a permanent buss number to the actual system buss number. The pattern is set to output to a specific buss number; this buss number is associated with a port name; this port name is looked up to see what system buss number is the one to be used. When the system setup changes, the pattern does not need to changed; only the port mapping may need to be changed. If the port map covers all I/O ports ever possible on a system, it will not need to be changed. See section 21 "Port Mapping" on page 227; it contains more details.

12.3.13 'rc' File / Keyboard Control Section

The keyboard control section has been merged into the MIDI control section and been moved into the 'ctrl' file. See section 12.5 "'ctrl' File" on page 163. There is no user-interface to change the keyboard control, for two reasons: (1) It is pretty easy to read, understand, and edit the 'ctrl' file, and (2) There are many more controls in *seq66*, and creating a user-interface to edit them might not be worth the effort. We suspect most users will be happy enough with the default settings, and

users of internationaly keyboards will find the 'ctrl' file easy enough to edit with a programmer's editor. As an example, see `qseq66-azerty.ctrl` in the `data/linux` directory.

12.3.14 'rc' File / JACK Transport

This section holds the settings for both JACK transport and for native JACK MIDI mode. The JACK Transport options are also command-line options. See section 12.2 "Command Line" on page 137. For *Seq66 before* 0.94.0, a number of boolean digit settings were used. For *Seq66 after* 0.94.0, the settings are more elegant.

```
[jack-transport]
transport-type = slave    # 'none', 'slave', 'master', or 'conditional'
song-start-mode = song    # 'song' = 'true' or 'live' = 'false'
jack-midi = true          # 'true' or 'false'
jack-auto-connect = true  # 'true' or 'false', default is true
```

'conditional' value tells *Seq66* to try to become JACK Master. If a master already exists, then *Seq66* falls back to JACK Slave. Note that JACK transport is separately configurable from JACK MIDI. Transport and MIDI use different JACK clients internally.

`song-start-mode` is set to 'true' or 'song' to force the usage of the track layout in the song editor (see section 7 "Song Editor" on page 100). If 'false' or 'live' then the live mode is in force, where the musician controls the arming of sequences. If 'auto', then the mode is set to 'song' if the loaded file has a track layout (triggers), and 'live' otherwise. It is probably best to leave it at 'auto'.

`jack-midi` set to 'true' turns on the usage of JACK for MIDI playback. If running in normal (not "manual/virtual" port) mode, then *Seq66* connects automatically to all JACK MIDI ports it finds on the system at startup.

`jack-auto-connect` is normally set to 'true'. If set to false, then the automatic connect discussed in the previous paragraph will not be performed. This option is useful if one want to rely on a session manager to make the connections, or do it onself, without using the manual/virtual port feature.

12.3.15 'rc' File / Reveal Ports

This option applies to both ALSA and JACK.

```
[reveal-ports]
show-system-ports = true   # flag for reveal ALSA ports
```

Turning on the reveal-ports option is necessary if one wants to see the actual port names defined by the system. It ignores the settings in the 'usr' configuration file's `user-midi-bus-definitions` and `user-midi-bus-N` sections. If this option is turned on, the definitions in the 'usr' configuration file are *not* read from that file.

12.3.16 'rc' File / Metronome

A very configurable metronome is supported. After running *Seq66* and then exiting, the following section is present:

```
[metronome]
output-buss = 1
output-channel = 0
beats-per-bar = 4
beat-width = 4
main-patch = 0
main-note = 72
main-note-velocity = 120
main-note-length = 0
sub-patch = 0
sub-note = 60
sub-note-velocity = 84
sub-note-length = 0
count-in-active = false
count-in-measures = 1
count-in-recording = true
recording-buss = 2
recording-measures = 0
```

The `main` settings apply to the first note of the metronome. The type of tone is given by the patch (program) number. The actual note is given by a MIDI note number. The velocity can be set. The note-length is controlled by a scale factor: 0 means an automatic calculation based on the beat width which equates to a value of 0.5; 1.0 means the note length is exactly equal to the beat width. The minimum is 0.125, and the maximum is 2.0. Try different values and see.

The `sub` settings apply to the rest of the notes of the metronome, but are otherwise the same. The settings `count-in-active` and `count-in-measures` control metronome count-in, where the metronome starts playing before the song. The rest of the settings are discussed in the following section on **Background Recording**.

There is a user-interface in **Edit / Preferences** to configure the settings of the metronome (and the recorder). See section 4.2.1.8 "Menu / Edit / Preferences / Metronome Options" on page 50.

12.3.17 'rc' File / Count-In Recording

Count-in recording allows recording to be made without having to create a pattern and turn on recording. It is useful when using a metronome and count-in measures. The following settings activate this feature and determine the input port to be used.

```
[metronome]
. . .
count-in-recording = true
recording-buss = 2
recording-measures = 0
```

The `count-in-recording` setting enables/disables count-in recording. The `recording-buss` setting specifies the input buss/port. That buss *must be enabled* (in the 'rc' file). The `recording-measures` sets the number of measures to record. The style of recording is currently *merge*, where notes accumulate as each loop occurs. We might adjust that feaure per user feedback. If `recording-measures` is set to 0, then the *expanding* mode of recording is used, where the count-in-recording sequence gets longer and longers as playback and playing continues.

12.3.18 'rc' File / Interaction Method

```
[interaction-method]
snap-split = false
click-edit = true
```

The `Mod4` ("Windows" key) option is no longer available, and not necessary. Also removed is the option for the "fruity" option of *Seq24*. It will be added back if there is a clamor for it.

This option comes from the *seq32* project. It allows for pattern-splitting in the Song editor at snap points, rather than just at the middle of the pattern.

This option allows one to enable/disable the ability to double-click in a pattern slot in the main window to bring it up for editing. This can interfere with a live performance where muting/unmuting come fast enough to be seen as a double-click.

12.3.19 'rc' File / Auto Option Save

This item determines if the 'rc' configuration file (and other files) is saved upon exit of *Seq66*. The normal behavior is to save it, which can sometimes be inconvenient when one is just trying out some command-line options.

```
[auto-option-save]
auto-save-rc = true
save-old-triggers = false
save-old-mutes = false
```

The 'save' options can be set to true to save triggers in the old formats. *Seq66* now saves triggers with a "transpose" value, so that clips can be transposed in the song editor for more extensive re-use (see the "Kraftwerk" MIDI file for an example). The 'mutes' are now save as a single byte, rather than a 4-byte value, to save some space.

12.3.20 'rc' File / Last Used Directory

The following item refers to the last directory in which one opened or saved a MIDI file.

```
[last-used-dir]
"/home/user/seq66/contrib/midi/"
```

It is used to select the initial directory in a save/open file dialog. It is updated whenever a MIDI file is loaded or saved, and hence causes a saving of the 'rc' file at exit of *Seq66*.

12.3.21 'rc' File / Recent Files

The following item preserves a list of the last few MIDI files loaded. It is not filled when a MIDI file is loaded via a play-list. The first number is the count of recent-files. The second number is a boolean to determine if the most-recent file should be loaded when *Seq66* starts. This option is useful as part of restoring a session.

```
[recent-files]
count = 2
load-most-recent = true
"/home/user/seq66-alternate/contrib/midi/2Bars.midi"
"contrib/midi/b4uacuse-seq24.midi"
```

12.3.22 'rc' File / Play-List

This item provides a configured set of named play-lists in a play-list file, and a flag to activate it. Having a playlist makes it easy to load song after song from pre-determined lists.

```
[playlist]
active = false
"/home/user/.config/seq66/sample.playlist"
```

See section 13 "Seq66 Play-Lists" on page 184. It describes the setup, layout, and usage of a *Seq66* playlist file containing one or more playlists.

12.4 'usr' File

This section describes the *Seq66* 'usr' (or "user") file. The main part of the *Seq66* 'usr' configuration file provides a way to give more informative names to the MIDI busses, MIDI channels, and MIDI controllers of a given system setup. This configuration overrides the default values of the **Event** drop-down list and the menu items in the pattern editor, and makes them reflect the names of the MIDI Control (CC) values of one's devices.

In *Seq66* it, also includes some items that affect the user-interface's look, and many other new configuration items.

After one runs *Seq66* for the first time (or after deleting the configuration files), it will generate a `qseq66.usr` file in one's "HOME" directory:

```
/home/user/.config/seq66/qseq66.usr             (Linux)
C:/Users/user/AppData/Local/seq66/qpseq66.usr   (Windows)
```

In a session manager, the files will be created in the session directory. Unlike the 'rc' file, the 'usr' file is *not* written every time *Seq66* exits. If the 'usr' files does not exist, one is created, but it is normally not overwritten thereafter. The important exception is if we have updated the 'usr' file with new option and have incremented the version number of this file. To cause it to be overwritten at exit, run *Seq66* with the `-u` or `--user-save` option:

```
$ qseq66 --user-save
```

This option is recommended when one installs a new version of *Seq66*, which might add new options to the 'usr' file. One usually must edit the 'usr' file manually. There are a few items that can be tweaked in **Edit / Preferences**, and, if modified, the user-save flag is turned on.

By default, the list of MIDI devices that *Seq66* shows depends on one's system setup and whether the manual-port option is specified or not. Here's our system, with the the `[manual-port]` option turned off, shown in a composite view with all menus one can look at for MIDI settings:

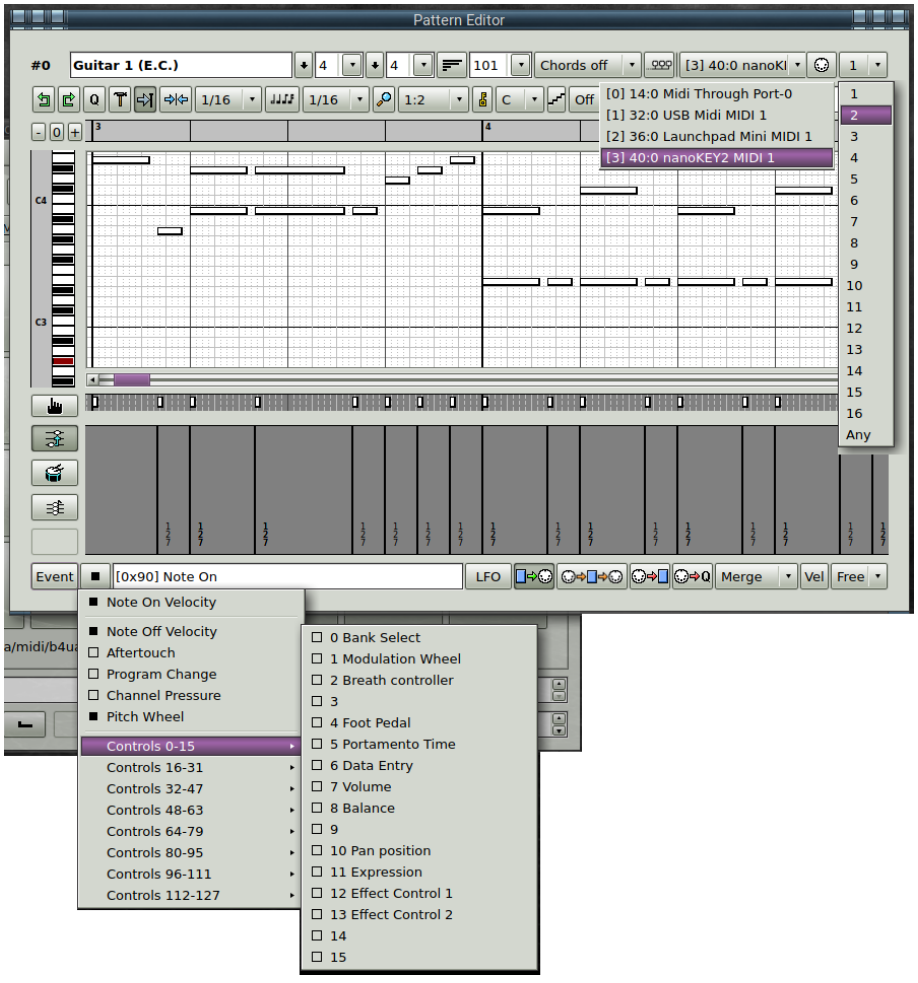


Figure 49: Seq66 Composite View of Native Devices

At the top, the buss dropdown menu contains the MIDI busses/ports active on this computer. At

right, the MIDI channel shows the channels numbers that can be picked for buss 0. At bottom left, we see the default controller values that *Seq66* includes. We have no idea if these correspond to the controllers that the selected MIDI device supports. We *can* use this dropdown to see if any such controller events are in the loaded MIDI file, of course; a solid black square indicates that such an event was found in the pattern.

To change the default lists, we can create sections for busses and instruments in the 'usr' file. The discussion here relies on the reader opening the file `sample.usr`, which is included in the shared `data/samples` directory provided once *Seq66* is installed.

Assume that we have 3 MIDI "buss" devices hooked to our system: two Model "2x2" MIDI port devices, and an old PCR-30 MIDI controller keyboard. Let's number them, using the convention that buss numbers and channel numbers start at 0, not 1:

- 0. Model 2x2 A
- 1. Model 2x2 B
- 2. PCR-30

Then assume that we have nine different MIDI instruments in our kit. let's number them, too:

- 0. Waldorf Micro Q
- 1. SuperNova
- 2. DrumStation
- 3. TX81Z
- 4. WaveStation
- 5. ESI-2000
- 6. ES-1
- 7. ER-1
- 8. TB-303

The *Waldorf Micro Q*, the *SuperNova*, and the *DrumStation* all have a large number of special MIDI controller values for modifying the sound they produce. The *DrumStation* accepts MIDI controllers that change various features of the sound of each type of drum it supports.

The buss devices shown here can be configured to route certain MIDI channels to certain MIDI devices. Assume we have them set up this way:

1. Bus 0: Model 2x2 A
    - SuperNova: channels 0 to 7
    - TX81Z: channels 8 to 10
    - Waldorf Micro Q: channels 11 to 14
    - DrumStation: channel 15
2. Bus 1: Model 2x2 B
    - WaveStation: channels 0 to 3
    - ESI-2000: channels 4 to 13
    - ES-1: channel 14

- ER-1: channel 15

3. Bus 2: PCR-30
   - TB-303: channel 0

We use the **'usr' configuration file**. to show these items with the proper names associated with each device, channel, and controller value The process for setting up the 'usr' file is to:

1. Define one or more MIDI busses, the name of each, and what instruments are on which channels. Each buss is configured in a section of the form "`[user-midi-bus-X]`", where "X" ranges from 0 on up. Each buss then defines up to 16 channel entries. Each entry includes the channel number and the number of a section in the user-instrument section described next.
2. Define all of the instruments and their controller names, if they have them. Each instrument is configured in a section of the form "`[user-instrument-X]`", where "X" ranges from 0 on up. Up to 128 controllers can be defined.

Let's walk through the structure of this setup, since it is a little bit tricky. Peruse the next couple of sections to understand a bit about the format of this file, following along in the sample 'usr' file.

12.4.1 'usr' File / MIDI Bus Definitions

This section begins with an "INI" group marker `[user-midi-bus-definitions]`. It defines the number of user busses that will be configured in this file. This section contains an number of `[user-midi-bus-N]` sections, where "N" ranges from 0 on upward. These correspond to the MIDI *output* busses expected to be in the system (ignoring the ALSA "announce" buss if present).

```
[user-midi-bus-definitions]
3     # number of user-defined MIDI busses
```

This means that the 'usr' file will have three MIDI buss sections: `[user-midi-bus-0]`, `[user-midi-bus-1]`, and `[user-midi-bus-2]`. Here's is an example of one such buss section:

```
[user-midi-bus-0]
2x2 A (SuperNova,Q,TX81Z,DrumStation)
16
0 1      # "channel" and "instrument number"
1 1      # Instrument #1 of the [user-instrument-definitions] section
. . .
8 3      # Instrument #3 of the [user-instrument-definitions] section
9 3
. . .
11 0     # Instrument #0 of the [user-instrument-definitions] section
12 0     # This is the Waldorf Micro Q device
. . .
15 2     # Instrument #2 of the [user-instrument-definitions] section
```

Each instrument is setup as a "channel" in a particular "buss". These instrument-definition sections are described in the next section. They are read from the 'usr' configuration file only if the "reveal

ports" option is *off* ("0"); this option can also be specified in the `[reveal-ports]` section of the 'rc' file. Otherwise, the actual port names reported by ALSA/JACK are shown.

The `user-midi-bus-definitions` and `user-midi-bus-N` sections can be misleading if one wants to have access to the actual MIDI port names that exist on the system. It is left as an exercise for the reader to try these different combinations of show-port options. Or one can consult the *Sequencer64 User Manual* to see the figures.

- Clocks View, -m (--manual-ports)
- Inputs View, -m (--manual-ports)
- Clocks View, -m (--manual-ports) and -R (--hide-ports)
- Clocks View, -r (--reveal-ports)
- Inputs View, -r (--reveal-ports)
- Clocks View, -R (--hide-ports)

12.4.2 'usr' File / MIDI Instrument Definitions

This section begins with an "INI" group marker `[user-instrument-definitions]`. It defines the number of user instruments that will be configured in this file. This section defines characteristics, such as the meanings of MIDI controller values, of the instruments themselves, not the MIDI busses to which they attached.

```
[user-instrument-definitions]
9     # number of user instruments
```

So this 'usr' file will define 9 instruments. We provide only one section as an example. Note that items without text default to the values prescribed by the General MIDI (GM) specification.

Each instrument contains up to 128 controller values; these controller values are available in the **Event** button in the Pattern Editor, and their names are shown.

```
[user-instrument-0]
Waldorf Micro Q                    # name of instrument
128                                # number of MIDI controllers
0                                  # first controller value, unnamed
1 Modulation Wheel
2 Breath Control
3
4 Foot Control
   . . .
123 All Notes Off (0)
124                                # defaults to GM
125 Unsupported
126 Unsupported
127                                # defaults to GM
```

Note the unnamed control numbers above. An unnamed control number might be an unsupported control number. It is termed to be "inactive". In this case, the **Event** menu of the Pattern editor

will show the default name of this controller. Again, though, the function denoted by this name might not be supported by the device. In that case, it might be better to call it "Unsupported". See the examples above. See the figure below for one example as set up using the `sample.usr` file:

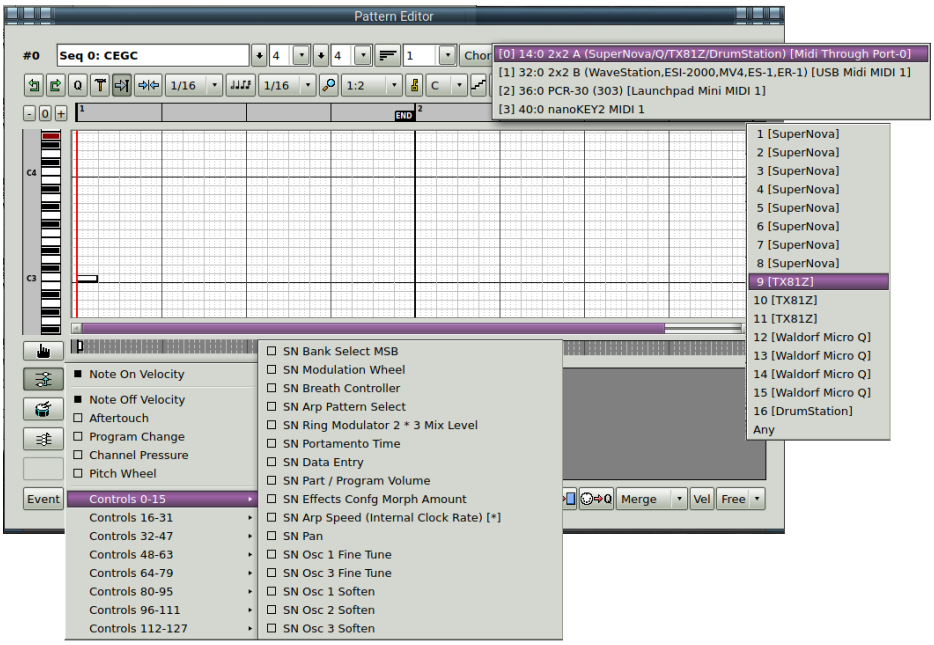


Figure 50: Seq66 Composite View of Devices As Set in "sample.usr"

12.4.3 'usr' File / User Interface Settings

This section, new to *Seq66*, begins with an "INI" group marker `[user-interface-settings]`.

It provides for a feature we will hopefully be able to complete some day: the complete specificition of the appearance of the user-interface. There is plenty of room to change the appearance of *Seq66* already. Please try the settings and see what looks good. Refer to either the sample file or the file generated when *Seq66* first runs.

```
[user-interface-settings]
swap-coordinates = false
mainwnd-rows = 4
mainwnd-columns = 8
mainwnd-spacing = 2
default-zoom = 2
global-seq-feature = true
progress-bar-thick = true
follow-progress = true
inverse-colors = false
time-fg-color = "lime"        # same as "default"
time-bg-color = "black"       # same as "default"
dark-theme = false
window-redraw-rate = 40
window-scale = 1
window-scale-y = 1
enable-learn-confirmation = true
```

`swap-coordinates` allows for an alternate mapping of pattern numbers. (Later, it will also apply to mappings for set-numbers and mute-group numbers.) Here is the pattern layout:

```
[  0 ] [  1 ] [  2 ] [  3 ] [  4 ] [  5 ] [  6 ] [  7 ]
[  8 ] [  9 ] [ 10 ] [ 11 ] [ 12 ] [ 13 ] [ 14 ] [ 15 ]
[ 16 ] [ 17 ] [ 18 ] [ 19 ] [ 20 ] [ 21 ] [ 22 ] [ 23 ]
[ 24 ] [ 25 ] [ 26 ] [ 27 ] [ 28 ] [ 29 ] [ 30 ] [ 31 ]
```

Compare it to the layout shown in section 5.1.3.1 "Pattern Keys" on page 68. Note that `progress-bar-thick`also applies to the following:

1. The vertical play-head in the pattern slots.
2. The vertical play-head in the pattern editor.
3. The vertical play-head in the song editor.
4. The font in the pattern slots (Live grid buttons) is made bold and made larger when this option is active.

There are a number of additional user-interface options. See the generated or sample 'usr' file for descriptions. Also see the chapter on palettes.

12.4.4 'usr' File / User MIDI PPQN

The long-standing PPQN for *Seq24* was a value of 192, and *Seq66* sticks with that default. This value is good for most tunes. But other sequencers allow for higher values. *Seq66* allows for some crazy values, ranging from 32 to 19200. If a MIDI file has a different PPQN, it will be rescaled to the default PPQN. However, one might want to stick with the PPQN specified in the MIDI file, so *Seq66* allows that as well. It is probably best to set `use-file-ppqn = true`, but that is up to the user.

```
[user-midi-ppqn]
default-ppqn = 192
use-file-ppqn = true
```

12.4.5 'usr' File / User Randomization

The **Jitter** and **Randomize** commands available in the **Tools** menu in the pattern editor depend on two configurable values.

```
[user-randomization]
jitter-divisor = 8
amplitude = 8
```

The `jitter-divisor` value is used to limit the amount of time jitter in jittering note events. If `J` is the jitter divisor, than the maximum range of time randomization R is: `R = S / J`, where `S` is the current value of note snap in the pattern editor. Thus, the time can be varied by an amount from minus R to plus R.

The `amplitude` value is the maximum range (plus or minus) by which to modify amplitude values such as note velocity or channel pressure. Keep this value small, as the numbers it affects range only from 0 to 127.

One minor bug persists in randomization. As the randomization is applied repeatedly, the amplitude tends toward zero.

12.4.6 'usr' File / User MIDI Settings

This section begins with an "INI" group marker `[user-midi-settings]`. It allows one to specify the global defaults for tempo, beats per measure, and so on.

```
[user-midi-settings]
convert-to-smf-1 = true
beats-per-bar = 4
beats-per-minute = 120
beat-width = 4
buss-override = -1
velocity-override = 80     # velocity_override (-1 = 'Free')
bpm-precision = 1          # 0, 1, or 2
bpm-step-increment = 0.1
bpm-page-increment = 5.0
bpm-minimum = 0
bpm-maximum = 127
```

The `convert-to-smf-1` option is normally true. This causes *Seq66* to convert single track MIDI files (in SMF 0 format) to multi-track SMF 1 files.

The `buss-override` option causes the buss-value (port number) to be applied to each pattern in a MIDI song that is loaded. This allows the tune to be directed to one's favorite synthesizer/application. Unlike the global port override in the main window (see section 3.2.3 "Buss Override for Play-set" on page 22), this application does not modify the file, though one can still opt to save it, which locks in the new buss number.

The `velocity-override` option fixes a long standing (from *Seq24*) bug where the actual incoming note velocity was always replaced by a hard-wired value. A value of "-1" corresponds to the "Free" setting, which preserves the incoming velocity.

The `bpm-precision`, `bpm-step-increment`, and `bpm-page-increment` values allow more precise control over tempo, which makes it easier to match the tempo of external music sources. Note that the step-increment is used by the up/down arrow buttons, the up/down arrow keys, and the MIDI BPM control values. The page-increment is used if the BPM field has focus and the Page-Up/Page-Down keys are pressed, and new MIDI control values have been added to support coarse MIDI control of tempo.

The `bpm-minimum` and `bpm-maximum` settings are used in scaling the display of Tempo events. By adjusting these values, one can more easily see the variations in tempo. In a main window pattern slot, or in the song editor tempo track, this range is scaled to the full range of note values, 0 to 127. Generally, one wants to select a range that keeps the main tempo line at the middle height of the pattern display.

To obtain these new settings, remember to backup the existing *seq66.usr*, then run *Seq66* with the `--user-save` option, and then do a "diff" on the new file and the original to merge any old values that need to be preserved. Then make any further tweaks to the new values.

12.4.7 'usr' File / User Options

This section begins with an "INI" group marker `[user-options]`. It provides for additional options keyed by the `-o`/`--option` options. This group of options serves to expand the options that are available, since *Seq66* is running out of single-character options. This group of options are shown below.

```
[user-options]
daemonize = false
log = "seq66.log"
pdf-viewer = "/usr/bin/zathura"
browser = "/usr/bin/lynx"
```

If this option is not used when running `seq66cli`, then the application stays in the console window and dumps informational output to it. If this option is in force, then the only way to affect `seq66cli` is to send a signal (e.g. SIGKILL) to it, or use MIDI control. One can also run *Seq66* `seq66cli` in the background from a console or from a desktop shortcut.

The log-file, if specified, is written to the same directory as the 'usr' file, the *Seq66* configuration directory. If empty, then a valid file-name can be specified in the `--option log=filename.log` option.

The PDF viewer and browser options are used if non-empty. Otherwise, the system default applications are used.

12.4.8 'usr' File / Additional Options

`[user-work-arounds]` is a section that is a relic from older versions of this application. It can be ignored. More useful options are described below.

12.4.8.1 'usr' File / Additional Options / [user-ui-tweaks]

`[user-ui-tweaks]` provides a small number of tweaks to the user-interface.

```
[user-ui-tweaks]
key-height = 10
note-resume = false
```

```
fingerprint-size = 128
progress-box-width = 0.8
progress-box-height = 0.4
progress-box-shown = true
progress-note-min = 20
progress-note-max = 100
lock-main-window = true
```

The `key-height` option affects the default "width" of the piano keys in the pattern sequence editor. Defaults to 12 (pixels). This option is also editable in the **Preferences** dialog. There are vertical zoom buttons, and the `v/0/V` keystrokes to change this on the fly, but those changes are not saved.

Seq66 can present the pattern editor in the 'Edit' tab versus an external window.. When used in the **Edit** tab instead of an external window, it is shrunk slight vertically to fit controls in the smaller window.

The following options adopt the new convention for setting variables, in which the format is `item-name = value`.

The `note-resume` option, if active, causes any notes in progress to be resumed when the pattern is toggled back on.

Note: the style-sheet option has been moved to the 'rc' file. See section 12.3.4 "'rc' File / Style Sheet" on page 143.

The `fingerprint` option provides a way to reduce the amount of drawing in the pattern grid. The pattern box in each button is small, and, for patterns longer than the fingerprint size, it makes no sense to draw hundreds of notes. Instead, patterns shorter than the fingerprint size are drawn normally, while longer patterns are drawn with a "fingerprint", a compressed representation of the pattern.

The `progress-box-width` and The `progress-box-height` options provide a way to expand or reduce the size of the progess box in each grid button. It is purely a user preference. The width ranges from 0.5 to 1.0 (full size), with a default of 0.8. The height ranges from 0.1 to 1.0 (full size), with a default 0f 0.3. Try setting both the width and height to 0.9 for an interesting effect.

If `progress-box-shown` is false, then the progress boxes are not drawn, and the pattern appears directly on the button.

The `progress-note-min` and The `progress-note-max` options change the note range in the progress box to control where in "pitch" the notes are shown.

The `lock-main-window` option, if true, prevents the main window from being resized. It can still be moved, and the external pattern and song editors can still be resized.

12.4.8.2 'usr' File / Additional Options / [user-session]

`[user-session]` provides a way to cooperate with the *Non Session Manager*. See section 9.3.3 "Session Management / NSM / Run with Remote NSM" on page 122; it goes into great detail.

```
[user-session]
```

```
session = none
url = ""
```

12.4.8.3 'usr' File / Additional Options / [new-pattern-editor]

`[new-pattern-editor]` contains settings values for recording when a new pattern editor is opened. A new pattern is indicated when the loop has the default name, *Unititled*. These values save time during a live recording session. The valid values for record-style are `merge`, `overwrite`, and `expand`.

```
[new-pattern-editor]
escape-pattern = false
armed = false
thru = false
record = false
qrecord = false
record-style = merge
wrap-around = false
```

The `escape-pattern` option applies to an open pattern window. If enabled, the `Esc` key can close the pattern window if patterns are not playing and if not in paint mode. If both are true, then the first `Esc` stops playing, the second `Esc` exits paint mode, and the third `Esc` closes the window. The use must enable this option deliberately.

Also included is a flag to allow notes to wrap around, where the Note On comes near the end of the pattern, but the Note Off appears before the Note On. In this case the end of the note is colored magenta. If wrap-around is false, then the note is clipped to the end of the pattern. **Warning**: If a song with wrapped notes is re-opened after wrap-around is changed to false, then the wrapped notes will be truncated. If one does not want to deal with wrap-around at all, then set the pattern length to the desired measure count *plus one* while recording, until satisfied with the recording. Shrink or delete any notes that bleed into the extra measure. Then set the pattern back to the desired length.

12.5 'ctrl' File

Seq66 provides a way to control the application to some extent via a MIDI controller, such as a MIDI keyboard or a MIDI pad. The current section describes this feature; additional resources and ideas can be found at `linuxaudio.org` and its discussion of MIDI control with *Seq24* [18]. Also see the tutorial section section 23 "Launchpad Mini" on page 236. An *Open Document Format* spreadsheet in the `doc` directory shows layouts for the default keystroke configuration (including pattern, mutes, and automation controls) and for a couple of *Launchpad Mini* configurations. Another spreadsheet lists the support names for keys; these names can be used in the 'ctrl' file, and include some changes for French AZERTY keyboards. The spreadsheets are `control_keys.ods` and `launchpad-mini.ods`.

The 'ctrl' file provides settings for keyboard control, MIDI control, and for specifying MIDI output to reflect *automation* commands in a device such as the *LaunchPad Mini*. The name of this file and its active status are specfied in the 'rc' file as noted earlier.

12.5.1 'ctrl' File / MIDI Control Settings

```
/home/user/.config/seq66/qseq66.usr             (Linux)
C:/Users/user/AppData/Local/seq66/qpseq66.usr   (Windows)
```

This file offloads the control settings from the 'rc' file, for a more flexible setup. It starts with the sections common to all *Seq66* configuration files. The first unique section defines some useful settings using the new variables feature of the configuration. Look at the sample or generated file to see the layout of these items.

```
[midi-control-settings]
control-buss = 3              # or 0xff
midi-enabled = true
button-offset = 0
button-rows = 4
button-columns = 8
keyboard-layout = qwerty
```

- `control-buss`. The control-buss value ranges from 0 to the maximum buss provided by the hardware on the system. If set, then only that buss will be allowed to send MIDI control. A value of 255 or 0xff means any buss can send MIDI control. If port-mapping is enabled, the short name (nick-name) of the port can be used as will.
- `midi-enabled`. If set to "true", then the MIDI controls will be used. It can be set to "false", while keeping the configuration in place for later usage.
- `button-offset`. This item provides a way to move a set of input controls (e.g. from a *Launchpad Mini*) to a different area of the input control device. Not yet supported.
- `button-rows`. Indicates the rows of the input control grid. Still in progress.
- `button-columns`. Indicates the columns of the input control grid. Still in progress.
- `keyboard-layout`. Provides a way to adapt to non-US keyboards. Currently, the only supported values are "normal" ("qwerty"), "qwertzy", and "azerty". For "azerty", the auto-shift feature of group-learning is disabled. The handling of keyboards can be quite complex, and differ between Linux distros and Windows. Especially problematic are the "dead keys" supported by some locales.

12.5.2 'ctrl' File / Loop Control

The loop-control group consists of 32 lines (0 to 31), one for each pattern slot shown in the patterns panel. It provides a way to control the arming/disarming (muting/unmuting) of each pattern shown in the patterns panel. It consolidates the keyboard and MIDI control settings into one table.

Note that the main window shows the *active* screen-set. These MIDI controls affect the *active* screen-set.

This block of matrix elements, numbered from 0 to 31, represent control functions (toggle, mute, unmute) for the 32 patterns of the active screen-set. These 32 rows correspond to the hot-keys assigned in the **File / Options / Keyboard / Control keys [keyboard-group]** configuration panel.

The MIDI control section begins with the following "INI"-style group marker tag, followed by one stanza-line per loop:

```
[loop-control]
# Control:  Toggle          On              Off
 0 "1"     [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Loop 0
 1 "q"     [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Loop 1
 . . .
31 ","     [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Loop 31
```

The first column is an index number, starting at 0. It indicates what loop the control line will affect. The second column is the name of the keystroke that will act as a toggle or action key. If the key name is `Blank`, then there is no keystroke for that pattern control. The user can specify multiple blank keys as desired.

The numbers in the leftmost brackets define a *Toggle* control; the numbers in the middle brackets define an *On* control; the numbers in the rightmost brackets define an *Off* control. The numbers inside each set of brackets define six values that set up the control. The layout of each filter inside the brackets is as follows:

[INV STAT D1 D2min D2max]

- **INV** = **inverse**
- **STAT** = **MIDI status byte** (channel included)
- **D1** = **data1**
- **D2min** = **data2 min**
- **D2max** = **data2 max**

If **STAT** is not 0x00, the control is enabled. *Seq66* will match the incoming MIDI event against the **STAT (MIDI status byte)** pattern (e.g. a Note On event), and perform the action (On/Off/Toggle) if the **D1** (e.g. a Note number), matches the incoming data, and the incoming parameters (e.g. Note velocity) falls in the specified **D2min** to **D2max** range. All data values are best specified in decimal.

The **INV (inverse)** field will make the pattern perform the opposite action (*off* for *on*, *on* for *off*) if the data falls outside the specified range. This is cool because one can map several sequences to a knob or fader.

The **STAT (MIDI status byte)** field is a MIDI status byte number in decimal or hexadecimal notation. Remember that it can include a channel. This channel is not overridden by the pattern's selected channel when a MIDI control matching event is received. One can look up the possible status values up in the MIDI messages tables; the relevant data can be found at *Summary of MIDI Messages* [19].

The last three fields describe the range of data that will match. The **D1 (data1)** field provides the actual MIDI event message number to detect, in decimal. This item could be a Note On/Off event or a Control/Mode change event, for example. The **D2min (data2 min)** field is the minimum value of the event for the filter to match. For Note On/Off events, this would be the velocity value, for example. The **D2max (data2 max)** field is the maximum value of the event for the filter to match.

For each pattern, we can set up MIDI events to turn a pattern on, off, or to toggle it. The loop MIDI control setup resembles a matrix. The default matrix uses the central keys on the keyboard, laid out in a 4 x 8 grid matching the pattern buttons:

```
1 2 3 4 5 6 7 8
q w e r t y u i
a s d f g h j k
z x c v b n m <
```

Please note that the buttons on some (or most) MIDI controllers send a message on press and another message on release. One can set such buttons up as toggle buttons by defining only the "Toggle" (first) stanza. Or one can have the button active the function on press, and then deactivate it on release by defining the "On" (second) and "Off" (third) stanzas appropriately.

12.5.3 'ctrl' File / Mute-Group Control

This section provides controls for 32 groups of mutes. A group is a set of patterns that can toggle their playing state together. Every group contains all 32 sequences in the active screen set. So, this part of the MIDI Control section is used for muting and unmuting (and toggling) a group of patterns using a keystroke or MIDI control. The definitions are in the same format as the loop-control section.

```
[mute-group-control]
  0 "!"  [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Mute 0
  1 "Q"  [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Mute 1
 . . .
 31 "<" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ]  # Mute 31
```

All this section does is set up the controls to be used; the actual mute-group patterns are defined in the 'mutes' configuration file (or in the *Seq66* MIDI file itself. A key name of `Blank` can be used to disable a keystroke for that line.

The mutes MIDI control setup resembles a matrix match the shifted versions of the loop control keys. The default matrix uses the central keys on the keyboard, laid out in a 4 x 8 grid matching the pattern buttons:

```
! @ # $ % ^ & *
Q W E R T Y U I
A S D F G H J K
Z X C V B N M <
```

12.5.4 'ctrl' File / Automation Control

This section provides ways to control *Seq66* push-button controls from a keyboard or from a MIDI device. These entries control *Seq66* actions like changing the BPM value, screen-set, record, solo, etc.

Note that automation controls that depend upon a parameter (such as group number or loop number) can only work with a MIDI control. MIDI control can provide parameters via a control value, note number or value, etc. Key control can provide only a "press" or "release" status.

Each item in this group consists of one line. Each line specifies a MIDI event that can cause a given *Seq66* user-interface operation to occur. These items are easy to view in the 'ctrl' configuration file, in the `[automation-control]` section.

```
[automation-control]
 0 "'"    [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # BPM Up
 1 ";"    [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # BPM Dn
 . . .
35 "Quit" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Loop 0
 . . .
48 "0xfe" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # Reserved 48
 . . .
80 "0x8f" [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] [ 0 0x00 0 0 0 ] # All Sets
```

One thing to be careful of when editing this section is to make all the numbers (from 0 to 80) unique. When copy/pasting a line and forgetting to change the number (*mea culpa*), an error like the following can appear in the console output:

```
Duplicate mute slot # 43 : '0xe8'
[seq66] Key controls: Error at line 232 ordinal 0xe8 key
   '0xe8' control 'Visibility' code 43
```

Note that "Quit" is not a real keystroke. It is a placeholder in the internal keystroke map for the functionality of quitting *Seq66* via MIDI control. Qt provides other ways to quit via keystroke. A key name of `Blank` can be used to disable a keystroke for a line. The stanzas meaning can change depending on the type of control. Here are the important styles:

```
Normal:        [ Toggle       ] [ On          ] [ Off             ]
Playback:      [ Pause Play   ] [ Start Play  ] [ Stop Play       ]
Play list:     [ Select by D2 ] [ Select Next ] [ Select Previous ]
Play song:     [ Select by D2 ] [ Select Next ] [ Select Previous ]
```

For selecting play-lists and songs by number, **D2** is used. Thus, one possible value to use to select would be to use a Note On event on channel 16 (0x9F) with a note number of 0 (a rarely-used note in any tune), and list/song numbers ranging from 0 to 127. Using note 0 for list selection and note 1 for song selection:

```
24 "F2" [ 0 0x9F   0   0 127 ] [ . . . ] [ . . .] # Play List
25 "F3" [ 0 0x9F   1   0 127 ] [ . . . ] [ . . .] # Play Song
```

Obviously, this requires a MIDI controller for which the velocity can be exactly specified. One can also reserve **D2** values of 126 for "previous" and 127 for "next".

12.5.4.1 Automation / BPM Up and Down

These controls increment or decrement the beats-per-minute setting, as if the up- or down-arrow has been clicked in the BPM combox-box, or the up- or down-arrow key pressed, in that combo-box. This increment is the "step increment" which defaults to 1, but can be modified by changing the "bpm_step_increment" value in the 'usr' configuration file.

12.5.4.2 Automation / Screen-Set Up and Down

Also abbreviated "Set Up" and "Set Down". This control increments / decrements to the next / previous screen-set. Once the screen-set has been altered, mute-groups and other actions apply to that screen set.

12.5.4.3 Automation / Mod Replace

This entry controls the "replace" flag. Once set, when the user manually clicks a pattern slot, that pattern is unmuted, and all the rest are muted. Thus, this MIDI control is kind a of "Solo" function. It works whether in "Live" or "Song" mode.

12.5.4.4 Automation / Mod Snaphot

This control causes the playing statuses of all active (i.e. having data) patterns to be saved. When turned off, the original playing status is restored.

12.5.4.5 Automation / Mod Queue

This control sets up the "queue" status flag. Then, when the user manually clicks a pattern slot, that pattern is queued, and will play at the next cycle of the pattern.

Here is an example from *Seq24* [18], which shows how to set up the "Sustain" control-change event to queue or un-queue a sequence: The *Akai MPK Mini* has a Sustain button and we can set the Sustain MIDI event (with MIDI status byte 176 [0xB0] to represent a Controller event, and control/mode change number 64 [0x40] to represent the Sustain or Pedal control) up as the queue modifier in the `mod queue` entry:

```
6 "o" [ 0  0x00 0  0  0 ] [ 0  0xB0 64 127 127 ] [ 0  0xB0 64 0  0 ]
#      INV STA  D1 mn mx   INV STA  D1 mn  mx     INV STA D1  mn mx
#                                  ^  ^                      ^  ^
#                                  |  |                      |  |
#                                  |   ----Sustain----------   |
#                                   -------Control Change-------
```

So when the Sustain button is held down, and one presses one of the pads on the *MPK Mini*, the corresponding sequence gets queued. Also included in the data directory are sample 'ctrl' files for other devices. For a more comprehensive discussion, see section 23 "Launchpad Mini" on page 236.

12.5.4.6 Automation / Mute Group ("Group Mute")

To be documented.

12.5.4.7 Automation / Group Learn

This MIDI control sets up a "group learn". This control sets two internal flags on : "mode-group" and "group-learn". The first flag indicates that we will be handling mute-groups. The second flag indicates that we are learning these mute-groups, effectively recording the current status of all the patterns in all of the screen-sets.

Note that this control corresponds to the "L" button in the main window user-interface. It can also be accessed by the default hot-key, `l`. Note that, once in learn-mode, there is no way to cancel learn-mode except by selecting an illegal mute-group keystroke. Also see section 15 "Seq66 Mutes Master" on page 195.

12.5.4.8 Automation / Playing Set

To be documented.

12.5.4.9 Automation / Playback

This automation entry defaults to a period. It starts playback and stops (pauses) playback. Note that this applies to the live grid. For the pattern and song editor piano rolls, it is hardwired.

12.5.4.10 Automation / Song Record

Initiates a "recording" of the musician's muting and unmutings as triggers in the song editor.

There are still some wrinkles to work out with this, such as ignoring snap values in order to get an exact recording of the triggers.

12.5.4.11 Automation / Solo

Meant to "solo" a given track. Needs testing and further work.

12.5.4.12 Automation / Theu

Turns on the "MIDI Thru" function of the current pattern as displayed in a pattern editor. More testing needed.

12.5.4.13 Automation / BPM Page Up and Page Down

Similar to section 12.5.4.1 "Automation / BPM Up and Down" on page 168, but in larger steps. These controls increment or decrement the beats-per-minute setting in large steps, as if the Page-Up

or Page-Down were pressed in the BPM combox-box. This increment is the "page increment" which defaults to 10, but can be modified by changing the "bpm_page_increment" value in the 'usr' configuration file.

12.5.4.14 Automation / Set a Set

Changes to a set as given by the data parameter. Needs more testing and further work.

12.5.4.15 Automation / Loop Mode

Toggles using the "L/R" markers as the beginning and ending of playback.

See section 12.5.4.29 "Automation / BBT/HMS and LR Loop" on page 172.

12.5.4.16 Automation / Quan Record

Toggles quantization of the incoming note events while recording.

12.5.4.17 Automation / Reset Sets

Resets the set counter to set 0.

12.5.4.18 Automation / One-shot

Toggles one-shot recording mode.

12.5.4.19 Automation / FF, Rewind, and Top

When activated, each command moves the playing tick value up or down by one-half of a measure. The "Top" command rewinds to the beginning immediately.

12.5.4.20 Automation / Playlist Commands

The "Play List" and "Play Song" automation commands allow one to select a particular play-list or song, or increment/decrement to the next/previous one.

12.5.4.21 Automation / Tap BPM

This command, when pressed repeatedly, sets the beats-per-minute value to the interval between the taps.

12.5.4.22 Automation / Start

This automation entry defaults to a space. It starts playback and stops playback with a rewind to the beginning. Note that this applies to the live grid. For the pattern and song editor piano rolls, it is hardwired.

12.5.4.23 Automation / Stop

This automation entry defaults to an escape character. It stops playback with a rewind to the beginning. Note that this applies to the live grid. For the pattern and song editor piano rolls, it is hardwired, and this keystroke can be used to exit insert/paint mode.

12.5.4.24 Automation / Toggle Mute

Reverses the armed statuses of the current set.

12.5.4.25 Automation / Song Position

This one needs work. We can't even remember what it means!

12.5.4.26 Automation / Keep Queuue

When given, this command turns on keep-queue mode.

12.5.4.27 Automation / Slot Shift

When given, this command allows for accessing patterns numbered from 32 to 63. When given again, this command allows for accessing patterns numbered from 64 to 95. Useful only when set sizes of 64 and 96 are configured.

12.5.4.28 Automation / Record Modes and Quantization

For release 0.97.3, **grid modes** have been introduced. The **Record** and **Quan Record** automation controls have been refactored to cycle through these modes. Additional automation commands provide direct access to each mode.

Grid mode changes the function of the main window's patterns panel so that it can be used to initiate recording instead of toggling mute status, regardless of whether the toggling is done via the buttons, hot-keys, or MIDI controls. The following modes are supported:

- **Loop**. The normal toggling of the mute/armed status of a pattern the slot is clicked or activated by its hot-keys.
- **Mutes**. Applies the given mute-group statuses when the slot is clicked. Since the "shifted" hot-keys can also be used, this mode is most useful with mouse-click control.

- **RECORD**. In this mode, a click on a slot or the use of its hot-key toggles recording for that pattern.
- **Copy**. Copies the pattern when that slot is clicked.
- **Paste**. Pastes the previously-copied pattern into the slot that is clicked.
- **Clear**. Clears the pattern in the clicked slot. The pattern remains in that slot, but it has no events. Careful!
- **Delete**. Deletes the pattern in the clicked slot. Careful!
- **Thru**. Enables the MIDI Thru function for that pattern.
- **Solo**. Solos the clicked pattern.
- **Cut**. Deletes the pattern in the clicked slot, while saving it in the clipboard.
- **Double**. Doubles the length (in measures) of the pattern in the slot that is clicked.

A click or a hot-key will cause the selected function above to be applied to the pattern denoted by the click/key. In **RECORD** mode, the following settings are enabled:

- **Overdub**. Called **Merge** in the pattern editor, this recording style just keeps adding events as they come in when the pattern repeats.
- **Overwrite**. When the pattern ends, then loops back to the beginning, an incoming note clears the pattern before being added.
- **Expand**. As notes come in, the pattern gets longer and longer.
- **One-shot**. Notes are added to the pattern as they come in, but recording stops when the end of the pattern's specified measures is reached.

Also supported is changing the mode of recording, that is, what happens to notes while incoming during recording:

- **No Quan**. Nothing is done to the incoming notes.
- **Quantize**. Incoming notes are quantized to the nearst snap value for the pattern.
- **Tighten**. Incoming notes are tightened (partially quantized) to the nearst snap value for the pattern.
- **Note-map**. If active, the specified 'drums' file values are used to remap the notes to new notes. This is useful, along with setting MIDI Thru, to play on a pre-General-MIDI drum machine (e.g. Yamaha DD-11) and hear it transmuted to GM while recording. Can also be applied after the fact, if the pattern is marked as transposable.

Also see section 6.8 "Pattern Editor / Bottom Row" on page 94for more information on these recording modes.

12.5.4.29 Automation / BBT/HMS and LR Loop

These commands toggle the display of bars:beats:ticks versus hours:minutes:seconds, and looping between the L/R markers.

See section 12.5.4.15 "Automation / Loop Mode" on page 170.

12.5.4.30 Automation / Undo and Redo

These commands undo or redo actions such as deleting notes. **Important**: These commands will affect *all* open pattern editors!

12.5.5 Automation / More MIDI Control

Many additional control items were requested by users, to control additional features of the application. Too many to list here. See the 'ctrl' file samples for more information. We will ultimately mention them all.

Some important ones to touch on:

- **Visibility**. Toggles the visibility of the user-interface. This can be useful in some circumstances. In addition, the session manager, if in force, might issue this command.
- **Quit**. Provides a way to exit *Seq66* via MIDI control.

12.5.6 'ctrl' File / MIDI Control Output

This section provides a way to have a MIDI device, such as the *Novation Launchpad Mini*, show the status of the patterns that are active, as well as other information. The first sub-section sets up some general settings.

```
[midi-control-out-settings]
set-size = 32
output-buss = 1
midi-enabled = true
button-offset = 0
button-rows = 8
button-columns = 8
```

- `set-size`. Provides the set size. The default is 32, in a 4 x 8 grid.
- `output-buss`. Indicates where automation-display controls are to be sent. Specify the output buss to which the display device is attached. If port-mapping is enabled, the short name (nick-name) of the port can be used.
- `midi-enabled`. If set to "true", then the MIDI control outputs will be used. It can be set to "false", while keeping the configuration in place for later usage.
- `button-offset`. This item provides a way to move a set of output controls (e.g. from a *Launchpad Mini*) to a different area of the output control device. Not yet supported.
- `button-rows`. Indicates the rows of the output control grid. Still in progress.
- `button-columns`. Indicates the columns of the output control grid. Still in progress.

```
[midi-control-out]
 0 [ 0x90  0 60 ] [ 0x90  0 15 ] [ 0x90  0 62 ] [ 0 0x90  0 12 ]
 1 [ 0x90  1 60 ] [ 0x90  1 15 ] [ 0x90  1 62 ] [ 0 0x90  1 12 ]
. . .
31 [ 0x90 31 60 ] [ 0x90 31 15 ] [ 0x90 31 62 ] [ 0 0x90 31 12 ]
```

The first number is the pattern number of the pattern whose armed/muted status is to be shown. There are samples in the `data/linux` directory for some devices that one can adapt to other equipment. The mute-group buttons and their status can also be shown:

```
[mute-control-out]
 0 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]
 1 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]
  . . .
31 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]
```

There are additional automation controls whose status can be displayed:

```
[automation-control-out]
0 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]  # Panic
0 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]  # Stop
 . . .
0 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]  # Tap_BPM
0 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]  # Quit
```

See the sample files for more detailed descriptions. Also see section 23 "Launchpad Mini" on page 236; it showns a fairly comprehensive setup based on the file `data/linux/qseq66-lp-mini.ctrl`.

12.5.7 'ctrl' File / Macro Control Output

This section provides a way to send setup information to a MIDI device, such as the *Novation Launchpad Pro Mk3*. There is too much involved (especially that device) to discuss in detail here, but this feature can be used to put a device into "programmer" mode at *Seq66* startup and back into "normal" mode at *Seq66* shutdown. Here is a hypothetical sample for the Mk3:

```
[macro-control-out]
footer = 0xf7
header =0xf0 0x00 0x20 0x29 0x02 0x0e
reset = $shutdown
shutdown = $live-mode
startup = $programmer-mode
live-mode = $header 0x0e 0x00 $footer
programmer-mode = $header 0x0e 0x01 $footer
```

The macros "footer", "header", "reset", "startup", and "shutdown" are written to the 'ctrl' file if not already present, though they won't have any useful definition. The first three are useful in other macros, while "startup" and "shutdown", if defined with actual data, are sent at the launch and shutdown, respectively, of *Seq66*. Note that these macros can be re-used in other macros, to increase the readability of the macros and to reduce the chance for mistakes. Note, too, that many devices required a device-specific SysEx header to precede the command-data, and the 0xf7 End-of-SysEx "footer" to end the command.

There is no limit to the number of macros that can be defined. To send any of them directly at any time, go to the **Session** tab and select a macro from the drop-down list.

There is currently no way to send them via MIDI control.

12.5.8 'ctrl' File / Keyboard / Default Assignments

This section provides a table of the functions, key numbers ("ordinals"), names, and other information about the default *Seq66* keyboard assignments. Also see the installed `control_keys.ods` spreadsheet, which might be more up-to-date.

The following status tags apply in this table. We're trying to support all keystrokes, but Qt and international keyboards make it sometimes difficult.

- **(X)**. Avoid using. Applies to the modifier keys Alt, Ctrl, Meta, Shift, etc. Also applies to tricky internal keys like the grave (backtick). One can try them, however, to see what happens.
- **(D)**. In the default configuration. Applies especially to the default loop-control and mute-group control keys that reside in the main part of the keyboard.
- **(d)**. In the default configuration, but no functionality yet.
- **(p)**. Available as a place-holder (a hex value)
- **(H)**. Hard-wired keys like Esc, Space, and the main arrow keys. Avoid using.
- **(A)**. Available for usage.
- **(!)**. Needs investigation.
- **(?)**. Needs investigation.
- *****. An asterisk added means "to do", as does the word "Reserved".

Because of the size of the table, and not wanting to deal with *LaTEX* long-table issues, we break the table into sections. These tables follow, some of them moved into following pages.

The first section is table 1 "Key Defaults. Control Keys" on page 176. Because of potential conflicts with user-interface keys, we do not recommend configuring them. However, we do use some of them for controls that are not yet effective. And, go ahead a try them if you want. The user rules!

The next section is table 2 "Key Defaults. ASCII Keys 1" on page 177. These deal with some of the pattern hot-keys ("Loop") and their shifted mute-group ("Mutes") keys. We do not show the numbers, as they are logically laid out on the U.S. keyboard. The Space and Period keys are hardwired ("*") for Start/Stop/Pause in the pattern piano roll and the song-editor piano roll.

The next section is table 3 "Key Defaults. ASCII Keys 2" on page 178. These deal mainly with the shifted mute-group ("Mutes") keys.

The next section is table 4 "Key Defaults. ASCII Keys 3" on page 179. These are mainly devoted to the "Loop" keys, which are laid out in a logical order on the keyboard.

The next section is table 5 "Key Defaults. Extended Keys 1" on page 180. Some of these keys (Esc and the arrow keys) are hardwired, as indicated by an asterisk ("*"). Do not redefine them. Also note the keys with hexadecimal number names (e.g. `0x88`). These are keys that we have not yet found mapped to a keystroke by the *Qt* keystroke system. Some of them are used as placeholders in the default key assignments of automation-control functions implemented only via MIDI control.

Table 1: Key Defaults. Control Keys

Function	Status	Ordinal	Name	Modifier
None	(X)	0x00	"NUL"	Ctrl
None	(X)	0x01	"SOH"	Ctrl
None	(X)	0x02	"STX"	Ctrl
None	(X)	0x03	"ETX"	Ctrl
None	(X)	0x04	"EOT"	Ctrl
None	(X)	0x05	"ENQ"	Ctrl
None	(X)	0x06	"ACK"	Ctrl
None	(X)	0x07	"BEL"	Ctrl
None	(!)	0x08	"BS"	Ctrl
None	(X)	0x09	"HT"	Ctrl
None	(!)	0x0a	"LF"	Ctrl
None	(X)	0x0b	"VT"	Ctrl
None	(X)	0x0c	"FF"	Ctrl
None	(X)	0x0d	"CR"	Ctrl
None	(X)	0x0e	"SO"	Ctrl
None	(X)	0x0f	"SI"	Ctrl
None	(X)	0x10	"DLE"	Ctrl
None	(X)	0x11	"DC1"	Ctrl
None	(X)	0x12	"DC2"	Ctrl
None	(X)	0x13	"DC3"	Ctrl
None	(X)	0x14	"DC4"	Ctrl
None	(X)	0x15	"NAK"	Ctrl
None	(X)	0x16	"SYN"	Ctrl
None	(X)	0x17	"ETB"	Ctrl
None	(X)	0x18	"CAN"	Ctrl
None	(X)	0x19	"EM"	Ctrl
None	(X)	0x1a	"SUB"	Ctrl
None	(X)	0x1b	"ESC"	Ctrl
None	(X)	0x1c	"FS"	Ctrl
None	(X)	0x1d	"GS"	Ctrl
None	(X)	0x1e	"RS"	Ctrl-Shift
None	(X)	0x1f	"US"	Ctrl-Shift

The next section is table 6 "Key Defaults. Extended Keys 2" on page 181. The main definitions here are for the "Function" and "Shift-Function" keys. Also note that one should avoid overriding the modifier keys. (But hey, see what happens!)

The next section is table 7 "Key Defaults. Extended Keys 3" on page 182. Note the some of the keypad keys are assigned, but there are many available. Presumably the keypad arrow keys are distinct from the main arrow keys, but that has not yet been tested.

The next section is table 8 "Key Defaults. Extended Keys 4" on page 183. There are many functions assigned in this section, but no real *Qt* keys defined. So this section is somewhat reserved for additional MIDI controls that will not have corresponding keystrokes. There are a lot more MIDI controls than keystrokes, especially leaving out the Ctrl, Shift, Alt, Super, and Hyper key

Table 2: Key Defaults. ASCII Keys 1

Function	Status	Ordinal	Name	Modifier
Start/Stop *	(H)	0x20	"Space"	none
Mutes	(D)	0x21	"!"	Shift
None	(A)	0x22	"¨	Shift
Mutes	(D)	0x23	"#"	Shift
Mutes	(D)	0x24	"$"	Shift
Mutes	(D)	0x25	"%"	Shift
Mutes	(D)	0x26	"&"	Shift
BPM Up	(D)	0x27	"'"	Shift
None	(A)	0x28	"("	Shift
None	(A)	0x29	")"	Shift
Mutes	(D)	0x2a	"*"	Shift
None	(A)	0x2b	"+"	Shift
None	(D)	0x2c	","	none
Event Edit	(D)	0x2d	"-"	Set-mode
Play/Pause *	(H)	0x2e	"."	none
Slot Shift	(H)	0x2f	"/"	none
Clear Mutes	(D)	0x30	"0"	none
Loop	(D)	0x31	"1"	none
Loop	(D)	0x32	"2"	none
Loop	(D)	0x33	"3"	none
Loop	(D)	0x34	"4"	none
Loop	(D)	0x35	"5"	none
Loop	(D)	0x36	"6"	none
Loop	(D)	0x37	"7"	none
Loop	(D)	0x38	"8"	none
None	(A)	0x39	"9"	none
None	(A)	0x3a	":"	Shift
BPM Down	(D)	0x3b	";"	none
Loop	(D)	0x3c	"<"	Shift
Pattern Edit	(D)	0x3d	"="	Set-mode
None	(A)	0x3e	">"	Shift
None	(A)	0x3f	"?"	Shift

combinations, which should generally be reserved for the operating system, window manager, and *Qt* user interface.

12.5.9 'ctrl' File / AZERTY and QWERTZ Keyboards

This section makes it clear how to adapt to an AZERTY or QWERTY keyboard. Keystrokes are mapped to loop, mute-group, and automation control in the `qseq66.ctrl` file in the `$HOME/.config/seq66` directory.

. **AZERTY**. Exit from `qseq66`, then copy the `qseq66-azerty.ctrl` file to the configuration/session directory. Make sure to edit `qseq66.rc` so that it has:

Table 3: Key Defaults. ASCII Keys 2

Function	Status	Ordinal	Name	Modifier
Mutes	(D)	0x40	"@"	Shift
Mutes	(D)	0x41	"A"	Shift
Mutes	(D)	0x42	"B"	Shift
Mutes	(D)	0x43	"C"	Shift
Mutes	(D)	0x44	"D"	Shift
Mutes	(D)	0x45	"E"	Shift
Mutes	(D)	0x46	"F"	Shift
Mutes	(D)	0x47	"G"	Shift
Mutes	(D)	0x48	"H"	Shift
Mutes	(D)	0x49	"I"	Shift
Mutes	(D)	0x4a	"J"	Shift
Mutes	(D)	0x4b	"K"	Shift
Mutes	(A)	0x4c	"L"	Shift
Mutes	(D)	0x4d	"M"	Shift
Mutes	(D)	0x4e	"N"	Shift
None	(A)	0x4f	"O"	Shift
Song Record	(D)	0x50	"P"	Shift
Mutes	(D)	0x51	"Q"	Shift
Mutes	(D)	0x52	"R"	Shift
Mutes	(D)	0x53	"S"	Shift
Mutes	(D)	0x54	"T"	Shift
Mutes	(D)	0x55	"U"	Shift
Mutes	(D)	0x56	"V"	Shift
Mutes	(D)	0x57	"W"	Shift
Mutes	(D)	0x58	"X"	Shift
Mutes	(D)	0x59	"Y"	Shift
Mutes	(D)	0x5a	"Z"	Shift
Screenset Down	(D)	0x5b	"["	none
Keep Queue	(D)	0x5c	"	
"	none			
Screenset Up	(D)	0x5d	"]"	none
Mutes	(D)	0x5e	"^"	Shift
Grid Mute Mode	(A)	0x5f	"_"	Shift
Group Mute	(D)	0x60	"`"	none

```
[midi-control-file]
active = true
name = "qseq66-azerty.ctrl"
```

Note that the `qseq66-azerty.ctrl` file specifies the following:

```
[midi-control-settings]
keyboard-layout = azerty
```

Table 4: Key Defaults. ASCII Keys 3

Function	**Status**	**Ordinal**	**Name**	**Modifier**
Loop	(D)	0x61	"a"	none
Loop	(D)	0x62	"b"	none
Loop	(D)	0x63	"c"	none
Loop	(D)	0x64	"d"	none
Loop	(D)	0x65	"e"	none
Loop	(D)	0x66	"f"	none
Loop	(D)	0x67	"g"	none
Loop	(D)	0x68	"h"	none
Loop	(D)	0x69	"i"	none
Loop	(D)	0x6a	"j"	none
Loop	(D)	0x6b	"k"	none
Group Learn	(D)	0x6c	"l"	none
Loop	(D)	0x6d	"m"	none
Loop	(D)	0x6e	"n"	none
Queue	(D)	0x6f	"o"	none
None	(A)	0x70	"p"	none
Loop	(D)	0x71	"q"	none
Loop	(D)	0x72	"r"	none
Loop	(D)	0x73	"s"	none
Loop	(D)	0x74	"t"	none
Loop	(D)	0x75	"u"	none
Loop	(D)	0x76	"v"	none
Loop	(D)	0x77	"w"	none
Loop	(D)	0x78	"x"	none
Loop	(D)	0x79	"y"	none
Loop	(D)	0x7a	"z"	none
None	(A)	0x7b	"{"	Shift
Oneshot Queue	(D)	0x7c	"|"	Shift
None	(A)	0x7d	"}"	Shift
Panic Button	(D)	0x7e	" "	Shift
None	(A)	0x7f	"DEL"	none

This code tells *Seq66* to (1) ignore the automatic shift-lock feature when learning mutes; and (2) slightly alters the internal key-map to place a few extended ASCII characters in it. This is necessary because of the extra keys and because one must use the `Shift` key to get the numbers on the numeric row of that keyboard layout.

. **QWERTZ**. We do not currently supply this layout, even though we support it, since it is quite similar to the default **QWERTY** layout; the user can easily edit it. Exit from `qseq66`, then copy the `qseq66.ctrl` file to `qseq66-qwertz.ctrl` (or a better name). Edit the latter to change "z" to "y", "y" to "z", "Z" to "Y", and "Y" to "Z". Make sure to edit `qseq66.rc` so that it has:

```
[midi-control-file]
"qseq66-qwertz.ctrl"
```

Table 5: Key Defaults. Extended Keys 1

Function	Status	Ordinal	Name	Modifier
Stop *	(H)	0x80	"Esc"	none
None	(A)	0x81	"Tab"	none
None	(A)	0x82	"BkTab"	Shift
Solo	(A)	0x83	"BkSpace"	none
None	(?)	0x84	"Return"	none
None	(?)	0x85	"Enter"	Keypad
Snapshot	(D)	0x86	"Ins"	none
None	(A)	0x87	"Del"	none
None	(p)	0x88	"0x88"	none
None	(p)	0x89	"0x89"	none
None	(X)	0x8a	"SysReq"	none
None	(X)	0x8b	"Clear"	none
None	(d)	0x8c	"0x8c"	none
None	(d)	0x8d	"0x8d"	none
None	(d)	0x8e	"0x8e"	none
None	(d)	0x8f	"0x8f"	none
Play Screenset	(D)	0x90	"Home"	none
None	(A)	0x91	"End"	none
Previous Song *	(H)	0x92	"Left"	none
Prev. Playlist *	(H)	0x93	"Up"	none
Next Song *	(H)	0x94	"Right"	none
Next Playlist .*	(H)	0x95	"Down"	none
BPM Page Up	(D)	0x96	"PageUp"	none
BPM Page Down	(D)	0x97	"PageDn"	none
None	(X)	0x98	"Shift_L"	Shift
None	(X)	0x99	"Ctrl_L"	Ctrl
None	(X)	0x9a	"Meta"	Meta
None	(X)	0x9b	"Alt_L"	Alt
None	(X)	0x9c	"CapsLk"	none
None	(X)	0x9d	"NumLk"	none
None	(X)	0x9e	"ScrlLk"	none
None	(p)	0x9f	"0x9f"	none

In the 'ctrl' file, make sure to specify the keyboard layout:

```
[midi-control-settings]
keyboard-layout = qwerty
```

Available layouts are "normal" (or "qwerty"), "qwertz", or "azerty". For now; more may be added as called for by users.

After starting and then exiting `qseq66`, the 'ctrl' file one has specified should still have the new settings. As usual, keep all your configurations in a safe place, such as a tar-file or ZIP-file.

One issue with some keyboard layouts are "dead keys". These keys do nothing but modify the next

Table 6: Key Defaults. Extended Keys 2

Function	Status	Ordinal	Name	Modifier
Top (beginning)	(D)	0xa0	"F1"	none
Next Playlist	(D)	0xa1	"F2"	none
Next Song	(D)	0xa2	"F3"	none
Follow Transport	(D)	0xa3	"F4"	none
Rew (depr)	(D)	0xa4	"F5"	none
FF (depr)	(D)	0xa5	"F6"	none
Song Pointer	(D)	0xa6	"F7"	none
Toggle Mutes	(D)	0xa7	"F8"	none
Tap BPM	(D)	0xa8	"F9"	none
Song/Live Mode	(D)	0xa9	"F10"	none
JACK Transport	(D)	0xaa	"F11"	none
Menu Mode (depr)	(D)	0xab	"F12"	none
None	(X)	0xac	"Super_L"	none
None	(X)	0xad	"Super_R"	none
None	(X)	0xae	"Menu"	none
None	(X)	0xaf	"Hyper_L"	none
None	(X)	0xb0	"Hyper_R"	none
None	(X)	0xb1	"Help"	none
None	(X)	0xb2	"Dir_L"	none
None	(X)	0xb3	"Dir_R"	none
Record Overdub	(D)	0xb4	"Sh_F1"	Shift
Record Overwrite	(D)	0xb5	"Sh_F2"	Shift
Record Expand	(D)	0xb6	"Sh_F3"	Shift
Record One-shot	(D)	0xb7	"Sh_F4"	Shift
Grid Loop Mode	(d)	0xb8	"Sh_F5"	Shift
Grid Record Mode	(d)	0xb9	"Sh_F6"	Shift-mode
Grid Copy Mode	(d)	0xba	"Sh_F7"	Shift-mode
Grid Paste Mode	(d)	0xbb	"Sh_F8"	Shift-mode
Grid Clear Mode	(d)	0xbc	"Sh_F9"	Shift-mode
Grid Delete Mode	(d)	0xbd	"Sh_F10"	Shift-mode
Grid Thru Mode	(d)	0xbe	"Sh_F11"	Shift-mode
Grid Solo Mode	(d)	0xbf	"Sh_F12"	Shift-mode
Reserved	(D)	0xc0	"KP_Ins"	Keypad

key that follows, and will not emit a useable key code. One will see some sample files with the extension 'keymap'. These files are not yet useful, but we anticipate calling them into play when more people are asking for support for their non-US keyboards.

12.6 'mutes' File

This file starts with:

```
[mute-group-flags]
load-mute-groups = midi       # load the mute-groups from MIDI file
```

Table 7: Key Defaults. Extended Keys 3

Function	Status	Ordinal	Name	Modifier
None	(A)	0xc1	"KP_Del"	Keypad
None	(X)	0xc2	"Pause"	Shift
None	(X)	0xc3	"Print"	Shift
Replace	(D)	0xc4	"KP_Home"	Keypad
None	(A)	0xc5	"KP_End"	Keypad
None	(?)	0xc6	"KP_Left"	Keypad
None	(?)	0xc7	"KP_Up"	Keypad
None	(?)	0xc8	"KP_Right"	Keypad
None	(?)	0xc9	"KP_Down"	Keypad
None	(A)	0xca	"KP_PageUp"	Keypad
None	(A)	0xcb	"KP_PageDn"	Keypad
None	(X)	0xcc	"KP_Begin"	none
None	(p)	0xcd	"0xcd"	none
None	(p)	0xce	"0xce"	none
None	(p)	0xcf	"0xcf"	none
Record Increment	(D)	0xd0	"KP_*"	Keypad
Reset Play-set	(D)	0xd1	"KP_+"	Keypad
None	(X)	0xd2	"KP_,"	Keypad
Quan Record Incr.	(D)	0xd3	"KP_-"	Keypad
Set Screenset	(D)	0xd4	"KP_."	Shift-Keypad
None	(A)	0xd5	"KP_/"	Keypad
None	(p)	0xd6	"0xd6"	none
None	(X)	0xd7	"Shift_R"	Shift
None	(X)	0xd8	"Ctrl_R"	Ctrl
None	(D)	0xd9	"KP_."	Keypad
None	(X)	0xda	"Alt_R"	Group
None	(X)	0xdb	"Shift_Lr"	none
None	(X)	0xdc	"Shift_Rr"	none
None	(X)	0xdd	"Ctrl_Lr"	none
None	(X)	0xde	"Ctrl_Rr"	none
Quit/Exit	(X)	0xdf	"Quit"	MIDI-control-only

```
save-mutes-to = both          # save to this file, MIDI file, or both
mute-group-rows = 4           # for now, stick with this value
mute-group-columns = 8        # for now, stick with this value
mute-group-selected = -1      # if 0 to 31, load that group at startup
groups-format = binary        # binary versus hex format for bits
toggle-active-only = false    # if true, toggle only mute-active patterns
```

These variables are described in the sample 'mutes' file. The mute-in group consists of 32 lines (32 to 63), one for each pattern box. It provides a way to control the mute groups. A group is a set of sequences that can arm their playing state together; every group contains all 32 sequences in the *active* screen-set.

This section is delimited by the `[mute-group]` construct. It controls 32 groups of mutes in the same

Table 8: Key Defaults. Extended Keys 4

Function	Status	Ordinal	Name	Modifier
Grid Veloc. Mode	(d)	0xe0	"0xe0"	none
Grid Double Mode	(d)	0xe1	"0xe1"	none
Grid Quant None	(d*)	0xe2	"0xe2"	none
Grid Quant Full	(d*)	0xe3	"0xe3"	none
Grid Quant Tight	(d*)	0xe4	"0xe4"	none
Grid Quant Random	(d*)	0xe5	"0xe5"	none
Grid Quant Jitter	(d*)	0xe6	"0xe6"	none
Grid Quant Reser.	(d)	0xe7	"0xe7"	none
BBT/HMS	(d*)	0xe8	"0xe8"	none
L/R Loop Mode	(d*)	0xe9	"0xe9"	none
Undo Record	(d*)	0xea	"0xea"	none
Redo Record	(d*)	0xeb	"0xeb"	none
Transpose Song	(d*)	0xec	"0xec"	none
Copy Set	(d*)	0xed	"0xed"	none
Paste Set	(d*)	0xee	"0xee"	none
Toggle Tracks	(d*)	0xef	"0xef"	none
Set Mode Normal	(p*)	0xf0	"0xf0"	none
Set Mode Auto	(p*)	0xf1	"0xf1"	none
Set Mode Adding	(p*)	0xf2	"0xf2"	none
Set Mode All	(p*)	0xf3	"0xf3"	none
None	(p)	0xf4	"0xf4"	none
None	(p)	0xf5	"0xf5"	none
None	(p)	0xf6	"0xf6"	none
None	(p)	0xf7	"0xf7"	none
None	(p)	0xf8	"0xf8"	none
Visibility	(D)	0xf9	"0xf9"	none
Save Session	(D)	0xfa	"0xfa"	none
Reserved	(D)	0xfb	"0xfb"	none
Reserved	(D)	0xfc	"0xfc"	none
Reserved	(D)	0xfd	"0xfd"	none
Reserved	(D)	0xfe	"0xfe"	none
Terminator	(X)	0xff	"Null_ff"	Illegal-value

way as defined for `[midi-control]`. A group is set of sequences that can toggle their playing state together. Every group contains all 32 sequences in the active screen set.

```
[mute-groups]
 0 [ 0 0 0 0 0 0 0 0 ] [ 0 0 0 0 0 0 0 0 ] [ . . .] [ 0 0 0 0 0 0 0 0 ]
 1 [ 0 0 0 0 0 0 0 0 ] [ 0 0 0 0 0 0 0 0 ] [ . . .] [ 0 0 0 0 0 0 0 0 ]
 . . .
31 [ 0 0 0 0 0 0 0 0 ] [ 0 0 0 0 0 0 0 0 ] [ . . .] [ 0 0 0 0 0 0 0 0 ]
```

In this group are the definitions of the state of the 32 (or more, once the support for larger sets is completely worked out) sequences in the playing screen set when a group is selected. Each set of

brackets defines a group.

12.7 'drums' File

The 'drums' file is based on a similar file created using the `midicvt` application (also available on *GitHub*. This file is also referred to as the 'note-mapper' file.

```
[notemap-flags]
map-type = drums
gm-channel = 10
reverse = false
```

These settings are explained in the sample 'drums' files. In addition, the file includes a number of sections that define the number and name of the original "drum", and the *General MIDI* device to which it corresponds.

```
[Drum 36]
dev-name = "Bass Drum Gated Reverb"
gm-name = "Bass Drum 1"
dev-note = 36
gm-note = 36
```

This file is useful mainly when obtaining drum tracks recorded with devices in the early days of MIDI, where each vendor provided their own peculiar layout of percussion sounds.

12.8 'palette' File

This file is described in the chapter on palettes, section 16 "Palettes for Coloring" on page 199.

12.9 'playlist' File

This file is described in the chapter on playlists, section 13 "Seq66 Play-Lists" on page 184.

13 Seq66 Play-Lists

Seq66 supports play-lists. A play-list provides a way to step through and play a number of MIDI files without having to load each one individually. The format of the play-list file is a variation on the 'rc' file, conventionally ending with the extension `.playlist`. It contains a number of play-lists, each described by a `[playlist]` section. Each play-list section provides a human-readable title which is selectable via a MIDI data number, or by moving to the next or previous playlist in the list using the `Up` and `Down` arrow keys. Each playlist section contains a list of MIDI files, also selectable via a MIDI data number, or by moving to the next or previous song in the list using the `Left` and `Right` arrow keys.

Movement between the playlists and the songs is accomplished via MIDI control or the arrow keys. Using MIDI control (`[midi-control]`, see section 12.5 "'ctrl' File" on page 163) makes it possible to use the `seq66cli` headless version of *Seq66* in a live setting. See section 23 "Launchpad Mini" on page 236; it describes programming for the *Novation LaunchPad Mini* for *Seq66*. In the normal user-interface, play-list movement can also be done manually via the four arrow keys on the computer keyboard.

The playlist file can be specified on the command-line, in the 'rc' file, or be loaded from the **File / Open Playlist** menu or in the **Playlist** tab. The playlist setup is written to the 'rc' file; the 'rc' file indicates the name of of the play-list file and if it is to be used or not.

The *Qt* user-interface supports editing of the play-list, using the **Playlist** tab. For version 0.99.7, this tab has been revamped. It was somewhat confusing to use, so the fixes make the process much easier.

The user can use a text editor to edit the play-list file, if careful. The play-list format is defined in the following section. Later sections describe the user-interface.

13.1 Seq66 Play-Lists / 'playlist' File Format

The play-list file, by convention, has a file-name of the form `sample.playlist`. The play-list file starts with a hardwired top banner. It can also have an optional comments section, much like the 'rc' and 'usr' files. It is *not* overwritten when *Seq66* exits, unless it has been modified in the **Playlist** tab.

```
[comments]
Comments added to this section are preserved....
```

A blank line (without even a space) ends the comment section. Following the comments section is a `[playlist-options]` section.

```
[playlist-options]
unmute-next-song = true   # the next song selection unmutes its patterns
auto-play = true          # if true, unmute and start play automatically
deep-verify = false       # If true, every MIDI song is opened and verified
```

The first option allows the load of the next song to unmute the patterns in that song. The second option causes the loaded song to start playing. This feature is useful for an unattended lengthy play-list. The thrid option causes each MIDI file to be opened to verify that it is an error-free play-list. This process can be time-consuming for large playlists. If set to false, *Seq66* still makes sure that each MIDI file in the play-list at least exists.

Following the options section are one or more `[playlist]` sections. Each represents a complete play-list. Here is the layout of a sample playlist section.

```
[playlist]

# Playlist number, arbitrary & unique. 0 to 127 recommended for MIDI control
```

```
number = 126
name = "Music for Serious Dogs"  # Display name of this play list
directory = "contrib/midi/"      # Storage directory for the tunes

# Provides the song-control number and the base file-name of each song in
# this playlist.  The playlist directory is used, unless the
# file-name contains a path.

70 "allofarow.mid"
71 "CountryStrum.midi"
72 "contrib/wrk/longhair.wrk"
```

A play-list file can have more than one `[playlist]` section. This allows for partitioning songs into various groups that can be easily selected (e.g. based on the mood of the musician or the audience).

After the `[playlist]` tag comes the play-list number. In order to use MIDI control to select the playlist, this number is limited to the range 0 to 127. If there is more than one `[playlist]` section, they are ordered by this number, regardless of where they sit in the play-list file.

Next comes a human-readable name for the playlist, which is meant to be displayed in the user-interface. If surrounded by quotes, the quotes are removed before usage.

Next is the song-storage directory. This directory is the default location in which to find the songs in that play-list. It can be an absolute directory or a relative directory. However, be wary of using relative directories, since they depend on where *Seq66* is run.

Note: If a song's file-name has its own directory component, that overrides the default song-storage directory.

Lastly, there is a list of MIDI song file-names, preceded by their numbers. As with the playlist numbers, they are restricted to the range of 0 to 127, for potential usage with MIDI control. The songs are ordered by this number, rather than by their position in the list.

13.2 Seq66 Play-Lists / 'rc' File

The most consistent way to specify a play-list is to add an entry like the following to the 'rc' file:

```
[playlist]
active = true
name = "/home/ahlstrom/.config/seq66/sample.playlist"
base-directory = ""
```

This setup allows a play-list file to be specified and activated. If the name of the play-list file does *not* contain a base-directory, then the play-list file is searched for in the user's *Seq66* configuration directory. If the play-list file-name is empty (i.e. set to `""`), then there is no play-list active.

13.3 Seq66 Play-Lists / 'ctrl' File / [midi-control]

The MIDI control stanzas for play-list and song-selection don't quite follow the toggle/on/off convention of the `[midi-control]` section, though the layout is the same:

```
   Key   Load-by-number     Next            Previous
24 "F2" [ 144 2 1 127 ] [ 144 4 1 127 ] [ 144 0 1 127 ] # Play List
25 "F3" [ 144 5 1 127 ] [ 144 3 1 127 ] [ 144 1 1 127 ] # Play Song
```

Note that the key assignments are actually unused here, but the four arrow keys are hard-wired for use in navigating the play-lists and songs.

Both lines specify setting the next playlist or song according to a MIDI data value, or via "next" and "previous" controls. The "next" and "previous" controls can be implemented by any MIDI event, including *Note On* or *Program Change*. However, the "value" section requires a MIDI event that provides a `d1` (second data byte) value, such as velocity, because this value is used as the MIDI control number to select a playlist or song item. So, the following setting,

```
24 "F2" [ 0x90  2  1 127] . . .
```

specifies that a *Note On* event with channel 0 (144 = 0x90) on note #2 with a velocity between the range 1 to 127 will select a play-list. However, this selection will be made only if the velocity ranges from 1 to 127, and there exists a selection with that velocity in the play-list file. This control requires a controller device that can be configured to provide the exact *Note On* event, including the exact velocity.

13.4 Seq66 Play-Lists / Command Line Invocation

The command-line options to specify (and activate) the play-list feature are:

```
-X playlistfile
--playlist playlistfile
```

The play-list file is either a base-name (e.g. `sample.playlist`) or a name that includes the full or partial path to the play-list file (e.g. `data/sample.playlist`). If no path is specified, the directory is the currently set *Seq66* configuration-file directory. For session (e.g. NSM) support, one must stick with the configuration directory; do not provide an explicit directory-name.

Please note that any play-list file specified on the command line, or loaded in the play-list user-interface, will be written into the 'rc' file's `[playlist]` section when *Seq66* exits.

13.5 Seq66 Play-Lists / Verification

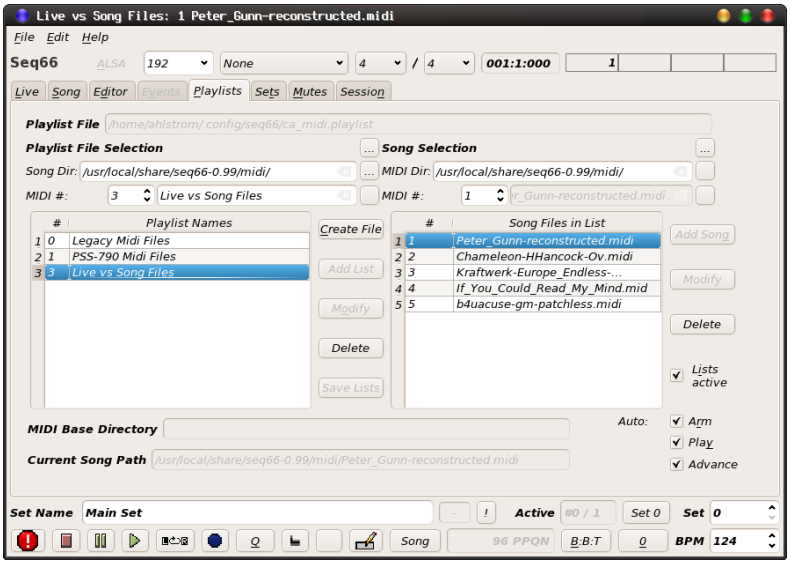
When *Seq66* loads a play-list file, the `deep-verify` option allows every song in the play-list file to be verified by loading it. If any load fails, then the playlist will fail to load. This check can be slow when there are many large MIDI files specified in the play-list file.

13.6 Seq66 Play-Lists / User Interface

Playlists and songs can be selected or moved-to via keystrokes or user-interface actions, in addition to MIDI control.

The `Up` and `Down` arrows move forward or backward through the list of play-lists, and the The `Right` and `Left` arrows move forward or backward through the list of songs for the currently-selected play-list.

The Qt 5 user-interface supports the display, selection, and editing of the play-lists and the song-list for each play-list. There are still some minor issues to work out. If encountered, close *Seq66* and edit the `.playlist` file manually. It is self-documenting.



Seq66 Playlist Tab

There is a lot to talk about in this tab. It has recently been modified to some extent, so note the changes.

Play-list specification:

1. **Playlist File**. This field displays the path to the loaded play-list file. It is not editable. Remember that a play-list file can contain multiple play-lists.
2. **Playlist File Selection**. These fields are editable, with the intent to use them to add a new play-list or modify the current one.
    1. **Selection**. This section has a button labelled **...** that replaces the old **Load List** button.

It opens a file-dialog to select an existing play-list file, reading it in, and logging the play-list directory sub-play-lists into the following items.

2. **Song Dir**. This field displays the main MIDI-file directory for the play-list currently selected in the table. The **Song Dir** is where the MIDI files reside by default.
3. **MIDI #**. This combo-box shows the MIDI control number for the current play-list, and the name of the selected play-list. Note that the **MIDI #** combo-box provides values from 0 to 127 which can be selected or edited. The entries are ordered by this number in the table. When a new play-list is specified, this number is set to one past the highest existing play-list number. It can be modified as long as it's not the same as another MIDI number.
4. **Name**. This field shows the user's name for the selected playlist.

3. **Playlist Names**. This table shows the MIDI-control number and the name of each play-list.
4. **List Buttons**. These buttons are described below. See section 13.6.1 "Seq66 Play-Lists / User Interfaces / Playlist Buttons" on page 189. Please note that, in some cases, the exact functionality might not be perfected.

Song specification:

1. **Song Selection**. These fields display the MIDI-file directory, the MIDI control number, and the file-name of the selected play-list.
   1. **Selection**. This section has a button labelled **...** that replaces the old **Load Song** button. It opens a file-dialog to select an existing MIDI file and logging the file's directory and file-name into the following items.
   2. **MIDI Dir**. This field displays the directory holding the selected MIDI file. Note that the directory is normally the play-list directory, but a path present in the MIDI file-name overrides that directory, and then an asterisk is shown to flag that status.
   3. **MIDI #**. The MIDI control number field is editable, with the intent to use it to add a new song. This combo-box shows the MIDI control number for the current MIDI file. Note that the **MIDI #** combo-box provides values from 0 to 127 which can be selected or edited. The entries are ordered by this number in the table. When a new MIDI file is specified, this number is set to one past the highest existing MIDI file number. It can be modified as long as it's not the same as another MIDI number.
   4. **Name**. Holds the name of the selected MIDI file. It is not editable because it needs to be an existing MIDI file. To assign the MIDI file a readable base name, give it a long file-name (such as `Live Play Version of Opus 11.midi`).
2. **Song Files in List**. This table shows the MIDI-control number and the name of each song.
3. **Song Files in List Buttons**. These buttons are described below. See section 13.6.2 "Seq66 Play-Lists / User Interfaces / Song Buttons" on page 190. Please note that, in some cases, the exact functionality is still being worked out or perfected.

13.6.1 Seq66 Play-Lists / User Interfaces / Playlist Buttons

This section briefly describes the "List" buttons to the right of the play-list table.

- **Create File**.
- **Add List**.
- **Modify**.

- **Delete**.
- **Save Lists**.

1. Create File. This button brings up file dialog; one selects the desired play-list directory and types in the name of the new play-list file. The file isn't actually created until **Save Lists** button is clicked.

2. Add. This button is enabled with the editing of the **Playlist Selection** fields. Once these three fields are correct, the new play-list can be added. The new play-list can then be populated with songs.

3. Modify. This button is enabled with the editing of the **Playlist Selection** fields. Once these three fields are correct, the list can be modified.

4. Delete. This button removes the currently-selected play-list from the play-list file. This action doesn't take effect until the play-list file is saved or *Seq66* exits and does its normal saving.

5. Save Lists. This button bring up a file dialog to save the current play-lists and songs into the specified play-list file.

13.6.2 Seq66 Play-Lists / User Interfaces / Song Buttons

This section briefly describes the "Song" buttons to the right of the song-list table.

- **Load Song**.
- **Add Song**.
- **Modify Song**.
- **Delete Song**.
- **Lists Active**.

1. Load Song. This button bring up a dialog to open a MIDI or WRK file from the current song-directory or from an arbitrary directory. Currently, be careful with this option; adding a file from an arbitrary directory will generally prepend that directory to the MIDI file-name, making the song list difficult to read.

2. Add Song. This button is meant to add a song already loaded in the **Live** frame into the play-list. Just open a new tune, test it, and then add it to the play list. Note that currently one may load a new tune into the playlist from anywhere a song is allowed to be loaded by the session.

3. Modify Song. This button is meant to modify song information. However, the only item that can be altered is the MIDI control number.

4. Delete Song. This button deletes the currently-selected song from the song list.

5. Lists Active. If checked, the play-list is enabled, and the arrow keys, automation keys, and MIDI controls (if configured) can be used to move between play-lists and songs. Note that this check-box will not work unless there are play-lists and songs that exist in the tables and on the computer's file system.

13.6.3 Seq66 Play-Lists / User Interfaces / Info Fields

The following read-only fields show some information about the file-system for the play-lists.

- **MIDI Base Directory**. Provides the top-most directory where all of the files in the play-list are stored. Currently read-only, in order not to interfere with session locations.
- **Current Song Path**. Shows the exact path the the currently-selected song. Currently read-only, in order not to interfere with session locations.

These items can be modified, however, by editing the play-list file directly.

In addition, the following check-boxes set a couple of play-list features.

- **Lists active**. This items reflects the status of the play-list being active as configured in the 'rc' file. When a new play-list file is created, this item can be check-marked if the play-lists and their songs are basically valid.
- **Auto: Arm**. This item sets the `unmute-next-song` (unfortunate name!) item in the 'playlist' file.
- **Auto: Play**. This item sets the `auto-play` item in the 'playlist' file. If set, the the **Arm** setting is automatically set as well. When set, the next song starts playing as soon as it is loaded.
- **Auto: Advance**. This item sets the `auto-advance` item in the 'playlist' file. If set, the the **Play** and **Arm** settings are automatically set as well. When set, at the end of one song, the next song is automatically loaded and played as well.

There is currently no `deep-verify` check-box; edit the 'playlist' file directly.

13.6.4 Seq66 Play-Lists / Creating a Playlist File

The "programmer's method" for creating a play-list file is simply to copy one and carefully modify it in a nice programmer's editor. This section describes creating a new play-list in the **Playlist** tab. We might add pictures are some point.

Let's assume this is the first run of *Seq66*. The file *qseq66.rc* is not created until after the first run exits. However, the configuration directory `~/.config/seq66/` is created at startup. Also, the default play-list file is set as

`~/.config/seq66/qseq66.playlist`. This file starts as a skeleton file, containing no play-lists.

But on our first run, we want to build a play-list, "from scratch", using the set of sample files files installed in `/usr/share/seq66-0.99/midi`. Start *Seq66* and select the **Playlist** tab. Note the **Playlist File** is set to `~/.config/seq66/qseq66.playlist`, and there are no play-lists and songs present. For this mini-tutorial we *ignore* the **Playlist File Selection** button (labelled **...**).

Click the **Create File** button. A file dialog comes up; it shows the default directory `~/.config/seq66/`, and the default file `qseq66.playlist`. For this tutorial, type in the file-name `tutorial.playlist`.

Click **Save** button. *Seq66* attempts to load the play-list file, but it does not yet exist. `~/.config/seq66/tutorial.p` is shown in the **Playlist File** field.

Now click the **Song Dir** button, and navigate to the `/usr/share/seq66-0.99/` directory and select the `midi` directory. Just select this directory, do not descend into it. Click **OK** or **Choose**. Change the play-list **Name** from "midi MIDI Files" to "Installed MIDI Files". The **Add List** button becomes enabled; click it.

Click on the **Song Selection** button. Navigate to where the desired play-list MIDI files are stored (the previously selected `midi` directory). Select the first desired MIDI file and click **OK** or **Open**. If it's the desired directory and file, then click the **Add Song** button. If the MIDI file's directory is *not* the same as the play-list's **Song Dir**, the song's **MIDI Dir** field holds the MIDI file's path, and there is an asterisk to indicate it's a MIDI-file-specific directory.

Add more songs and play-lists as desired. Note that the **MIDI** # number self-increments, but these can later be modified. Next, click **Lists active** to enable the new lists. Click **Auto arm** if desired. Click **Save Lists**. If desired, open the new play-list file in a text editor and see what it looks like.

13.6.5 Seq66 Play-Lists / Playlist Sample

This section describes the sample play-list that accesses the MIDI data files installed in `/usr/share/seq66-0.99/mid`

TODO

14 Seq66 Set Master

The **Set Master** is a way to get a global view of all the screensets in a *Seq66* MIDI file, and to be able to do some simple operations (movement, naming, etc.) with the sets. In the latest version of *Seq66*, there are always 32 sets. This simplifies the handling of sets. Some sets may be empty.

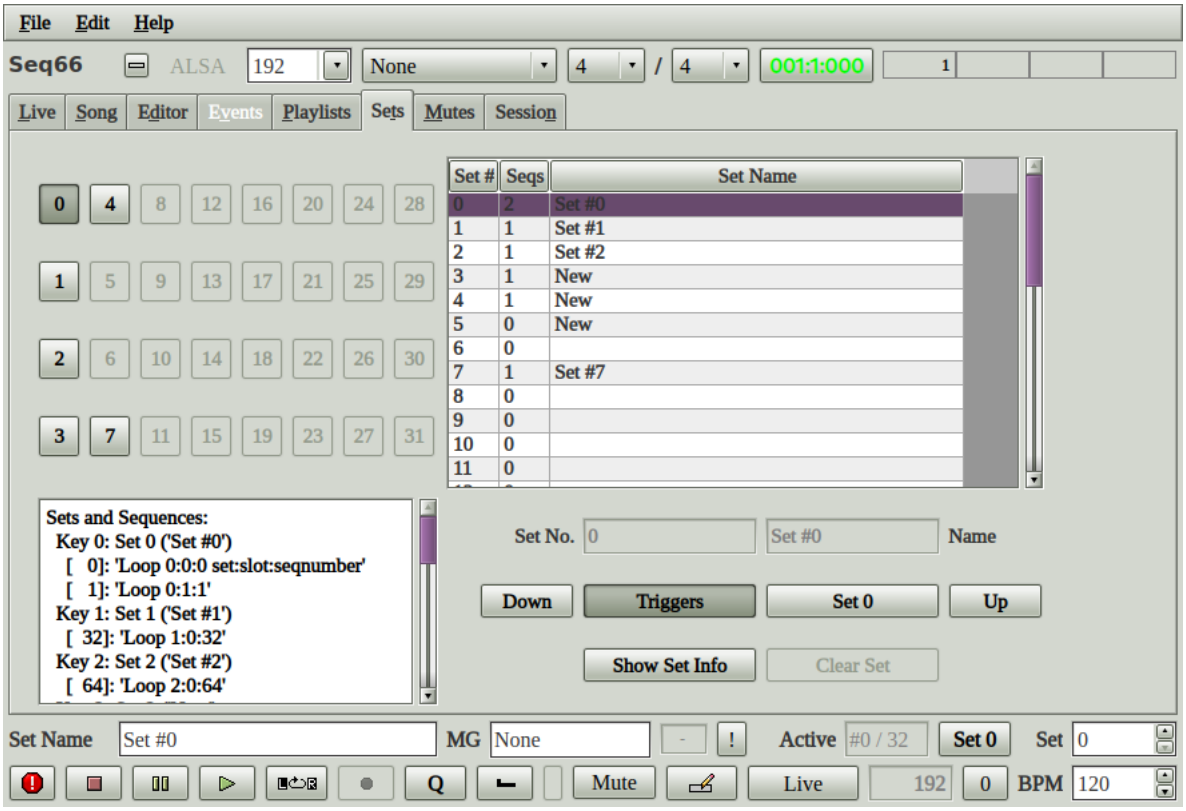


Figure 51: Sets Tab

The operations that can be done consist of viewing the sets, making a screenset active, rearranging the sets, clearing sets, and getting a survey of the contents of the sets.

1. Sets Grid. The set grid is always 4 x 8. Like the mute-groups, there's not a lot of benefit supporting more sets. The reader may wish to email us arguing for a different point of view. This grid shows the non-empty sets that are present in the MIDI file, represented by enabled buttons. With **Triggers** mode set, this allows one to choose which set is currently active (i.e. which is the "play screen"). Click on it and it is active. Also remember that keystrokes (`[` and `]` by default) in the main window can move the playscreen to the previous or next set.

2. Set List. The table at the right shows the set numbers, how many patterns/sequences are in each set, and the set name. The set name is editable once it is double-clicked.

3. Set Number/Name Fields. Currently just shows the number and name of the selected set. Eventually, these fields will be editable.

4. Up/Down Buttons. These buttons allow the user to move the selected set up and down in the list.

5. Show Set Info. Clicking this button lists all the sets and their tracks in the read-only edit box at the left.

6. Clear Set. Clicking this button deletes the patterns from the set selected in the table. Note that the 0th set cannot be cleared. Would one ever want to do that? In *Seq66*, there must always

be a set 0.

7. Triggers. This checkable button toggles the action done by clicking a button in the set grid. If checked, then clicking an active set button will make that set active during playback. That effect depends on whether the sets are configured to auto-arm when selected, or not. If not checked, then clicking an active set button merely summarizes the patterns in the set, in the text field below the set grid.

8. Set 0. If trigger mode is in effect, clicking this button makes set 0 active. An alternative to clicking on grid button 0.

14.1 Set Handling

This section talks about how sets work. The topics are

- **Set Management**
- **Empty-Set Handling**

14.1.1 Set Management

This section will discuss the work-flows of using sets to organize a song and to control playback. MORE TO COME.

14.1.2 Empty Set Handling

When *Seq66* loads a song, it loads the existing sets in the song and sets their names as stored in the `c_notes` *SeqSpec* (see section 25.2 "Sequencer-Specific Meta-Events Format" on page 251). In addition, one dummy and invisible set is created for internal management purposes.

When a new song is created, one usable set, Set #0, is always created, as a starting point. One generally starts with this set and adds patterns to it.

When one selects the next set (e.g. using the **Live** frame's **Set** spin control), that set does not exist, but is immediately created. So now the song has two sets, with the second one being empty. If the song is now saved, so is the empty set's file name. However, empty sets are not saved; a set must be populated with at least one pattern to be saved.

The following figure shows what happens when a song with 4 sets (0, 1, 2, and 7) is loaded, and then the user increments the spin-button all the way to set 8.

Figure 52: New Sets Creation

There are new sets 3, 4, 5, 6, and 8. However, if one saves and then reloads this song, the empty sets are gone. Just something to be aware of.

15 Seq66 Mutes Master

The **Mutes** tab provides a global view of all the mute-groups in a *Seq66* MIDI file or global configuration. It also provides for manipulation of mute-groups, to supplement what can be done via the live grid..

15.1 Mute Group

A mute-group (see section 24.2.8 "Concepts / Terms / group, mute-group" on page 249) holds the desired statuses (armed/muted) of all of the patterns in a screenset/bank (see section 24.2.3 "Concepts / Terms / bank, screenset, play-screen" on page 248). By setting up multiple mute-groups, one can quickly transition to different arrangements or musical themes.

A mute-group can be associated with a hot-key or MIDI control; when the mute-group is selected, the patterns that are armed changes. This makes it easy to change what the tune is playing in a big way. Mute-groups are *learned* by

1. Arming all of the desired patterns in the set.

2. Clicking the **Learn** ("L") button. See section 3.5.9 "Mute Group Learn" on page 27.
3. Clicking the keystroke, which is the shifted version of the hot-key to arm/mute the pattern corresponding to the mute-group number.. This is accomplished in the default keyboard configuration via the auto-shift function, which shifts the keystroke automatically for group-learn.

These steps can also be done via MIDI control. More tediously, mute-groups can be edited in a 'mutes' file.

To learn more about mute-groups, see section 3.5.9 "Mute Group Learn" on page 27, and section 24.2.8 "Concepts / Terms / group, mute-group" on page 249.

15.2 Mute Master Tab

By themselves, mute groups are not easy to see except by activating each mute-group. The mute-master (**Mutes** tab provides a way to view them more systematically, as well as assigning names to them.

Figure 53: Mute Master Tab

This diagram shows the **Mutes** tab after some mute-groups have been created. Mute-groups can be created in the main window's patterns panel, but it is difficult to know what each group comprises. This tab makes it easy to see the layout of the mute-groups, and also allows for some editing and live control of the mute-groups.

Note that the Qt theme shown here (*kvantum*) shows that buttons are checkable; other themes often do not show that status. Annoying.

15.3 Mute-Groups Table

The most important user-interface element is the table at the right. Here, one can select a mute-group for editing and control.

1. Mute-Groups Table. At the right is a table that shows the assigned mute-group keys and some information about them:

- **Group**. This column holds the group numbers for each group, ranging from 0 to 31. Each row corresponds to a button in the **Mute-Groups** grid.
- **Active**. This column shows the number of activated patterns in the mute-group. A zero means the mute-group is inactive.
- **Key**. Indicates the keystroke that can be used to activate that mute-group. By default, these are shifted version of the corresponding mute/unmute pattern-slot hotkey.
- **Group Name**. Provides a mnemonic name for the mute group. A feature for the future.

Only the group-name column of This table is editable. The user can also edit `qseq66.mutes` with a text editor, or edit `qseq66.ctrl` to modify the the hot-keys or MIDI controls.

Click on the desired group.

15.4 Mute-Groups Buttons

Once a mute-group has been selected, its mute-group button is activated. Click on the group button (perhaps twice), to be able to add pattern mute states via the pattern buttons, as an alternative to group-learn.

2. Mute-Groups. This grid is always of size **4 x 8**. It represents the maximum of 32 mute-groups that can be supported by *Seq66*. To start, all group buttons are *disabled* and *unchecked* (inactive). Where a mute-group exists, the button is made *checked* (active), but still disabled.

Here, the user clicked on mute-group 2, which becomes active in the user-interface. The **2** button is also enabled, and can be clicked.

Clicking once deactivates the button, which potentially flags that mute-group for removal. Clicking it again reactivates it and enables all of the buttons in the **Group Patterns** grid.

Also, if the **Triggers** button has been activated, then these buttons can activate mute-groups, as if activated in the live grid.

15.5 Group-Patterns Buttons

3. Group Patterns. Once this grid is enabled by clicking on a mute-groups button, each button here can be clicked to add a pattern to the mute-group, or remove a pattern from the mute-group.

15.6 Other Controls

4. Mutes File. The mutes-file read-only field shows the base-name of a file into which one can save the current-mute group setup, as a way to back up the setup. To change this file-name, do so in **Edit** / **Preferences** / **Session** / **.mutes**.

5. Save File. This button saves all of the mute-groups to the specified 'mutes' file. It is enabled when any change has been made to a mute-group or to the check-boxes. If the user has provided a path in the **Edit** / **Preferences** / **Session** / **.mutes** field, the path is stripped; *Seq66* must not write configuration information outside of the session configuration directory.

6. Load File.

This button brings up a dialog to select an external mutes file.

7. Read/Write Format. This section provides the following features, which still need some work:

- **From MIDI**. If checked, the mute-groups are read from the MIDI file, if present. These mute-group override the one from an external 'mutes' file.
- **From Mutes**. If checked, the mute-groups are read from the specified 'mutes' file, if present and active.
- **Binary**. This flag indicates to save the mute-group information in binary format, which is the normal format. Each mute-group pattern's setting is indicated by a 0 or a 1. This is the default format for writing the mute-groups.
- **Hex**. In this format, each set of mute-group is written in 8-bit hexadecimal format (e.g. "0xff"). This format is useful if the user has opted to have large set sizes such as 64 and 96 patterns. Not well-supported yet.
- **To MIDI**. When a tune is closed, as when *Seq66* exits, this option indicates to write the mute-group information to the *Seq66*-style MIDI file.
- **To Mutes**. When a tune is closed, as when *Seq66* exits, this option indicates to write the mute-group information to the 'mutes' configuration file (e.g. `qseq66.mutes`). Mute-groups can be written to both.
- **Toggle Only Active**. Normally, if a musician turns on a mute-group, then toggles other patterns separately, all patterns are turned off when the musician selects another mute group. If this option is checked, only the patterns that are part of the mute-group are toggled; the other patterns remain on.
- **Strip If No Mutes**. If checked, and there are no defined mute-groups, then they are not saved to the song.

The only time one would want to read/write the mute-groups to a 'mutes' file is when one uses standard song sets for all of one's songs.

8. Triggers. When activated, this option will enable the **Mute-Groups** buttons, deactive them all, and turn them into standard push-buttons. in the **Triggers** mode, when a button is clicked, the corresponding mute-group will be actived.

9. Zeroes. This button will clear every mute group, and replace them with mute-group where all patterns are off (zeroed).

There are a couple of small buttons. The first is reserved for expansion. The second displays an asterisk when a mute-group or mute setting has changed.

16 Palettes for Coloring

Many user-interface elements in *Seq66* are drawn independently of the Qt theme in force, and they have their own coloring. Also, patterns can be colored, and the color is stored (as a color number) in the pattern when the tune is saved. There are four palettes:

- **Pattern**. This palette contains 32 color entries, and each can be used to add color to a pattern in the *Live* grid or in the *Song* editor. The color of a pattern, if used, is saved with the pattern in the MIDI file.
- **Ui**. This palette contains 24 color entries. These color entries are used in drawing text, backgrounds, grid lines, background patterns, drum notes, and more. These colors each have a counterpart that is used with the `--inverse` option is applied to a run of *Seq66*.
- **Inverse Ui**. This palette contains 24 color entries. These colors are used when the `--inverse` option is applied to a run of *Seq66*.
- **Brushes**. This "palette" provides a way to specify the fill type for the drawing of notes, the scale (if shown) in the pattern editor, and the background sequence (if shown). It allows the user to select solid file, hatching, and some other fill patterns.

All palettes have default values built into the application. However, the user can also include 'palette' files to change the colors used. For example, the normal colored palette can be changed to a gray-scale palette. The name of the palette file is specified in the 'rc' file by lines like the following:

```
[palette-file]
1     # palette_active
qseq66-alt-gray.palette
```

If this palette file is active, it is loaded, changing all of the palettes, and thus the coloring of *Seq66*. In section 16.3 "Theming" on page 201, some weird behavior with the `qt5ct` configuration application and non-Qt-based window managers is discussed.

16.1 Palettes Setup

The palette file is a standard *Seq66* configuration file with a name something like `qseq66.palette`, plus three sections:

```
[palette]
[ui-palette]
[brushes]
```

The first section is the "Pattern" palette; the second section is the "Ui" palette, which includes

the inverse palette as well; and the third section defines brushes for drawing the interiors of some elements.

16.1.1 Palettes Setup / Pattern

The pattern palette is meant for changing the color of the progress area of the grid slot. This coloring helps in locating a pattern in the grid at a glance. The color selected for a pattern is also shown in the Song editor; both the name of the pattern and any triggers associated with the pattern are painted in the same color.

Each color is numbered. When a color is applied to the pattern, that color number is stored in a SeqSpec in the pattern, and saved in the MIDI file.

The following shows the pattern palette, with some entries elided for brevity:

```
      [palette]
       0            "Black" [ 0xFF000000 ]      "White" [ 0xFFFFFFFF ]
       1              "Red" [ 0xFFFF0000 ]      "White" [ 0xFFFFFFFF ]
       2            "Green" [ 0xFF008000 ]      "White" [ 0xFFFFFFFF ]
       3           "Yellow" [ 0xFFFFFF00 ]      "Black" [ 0xFF000000 ]
       4             "Blue" [ 0xFF0000FF ]      "White" [ 0xFFFFFFFF ]
       ...            ...       ...             ...       ...
      29      "Dark Violet" [ 0xFF9400D3 ]      "Black" [ 0xFF000000 ]
      30       "Light Grey" [ 0xFF778899 ]      "Black" [ 0xFF000000 ]
      31        "Dark Grey" [ 0xFF2F4F4F ]      "Black" [ 0xFF000000 ]
project.
```

The names are color names, and these names are what show up in the popup color menus for the pattern buttons in the *Live* grid. The colors on the left are the background colors, and the colors on the right are the foreground colors, which are chosen for contrast with the background. The colors are in `#AARRGGB` format, with the "#" replaced by "0x" because "#" starts a comment in *Seq66* configuration files. Note that all the alpha values are "FF" (opaque); we have not yet experimented with changing them. Lastly, only 32 entries are accepted.

16.1.2 Palettes Setup / Ui and Inverse Ui

The "UI palette" applies to other elements drawn by *Seq66*: foreground lines, background paint that can differ between various backgrounds, text colors that can differ between slot, time-panels, the data panel, and the Song editor's name panel.

If the user has chosen a particular Qt theme or has applied a style-sheet, a palette can be created to match. There are a couple of style-sheet/palette pairs in the `data/samples` directory.

The following shows the pattern and song palette, with some entries elided for brevity:

```
      [ui-palette]
       0       "Foreground" [ 0xFF000000 ]  "Foreground" [ 0xFFFFFFFF ]
       1       "Background" [ 0xFFFFFFFF ]  "Background" [ 0xFF000000 ]
       2            "Label" [ 0xFF000000 ]       "Label" [ 0xFFFFFFFF ]
```

```
 3        "Selection" [ 0xFFFFA500 ]   "Selection" [ 0xFFFF00FF ]
 4             "Drum" [ 0xFFFF0000 ]        "Drum" [ 0xFF000080 ]
       ...            ...       ...         ...       ...
29       "Slots Text" [ 0xFF000000 ]  "Slots Text" [ 0xFFFFFFFF ]
30          "Extra 1" [ 0xFF000000 ]     "Extra 1" [ 0xFFFFFFFF ]
31          "Extra 2" [ 0xFF000000 ]     "Extra 2" [ 0xFFFFFFFF ]
```

Here, the names are feature names, not color names. The first color is the normal color, and the second color is the inverse color. Only 32 entries are supported. The numbers have no meaning except to order the colors.

16.1.3 Palettes Setup / Brushes

The last "palette" is small, allowing the fill-pattern of a few pattern-editor items to be changed.

```
[brushes]
empty = solid              # preferred
note = lineargradient      # default
scale = dense3
backseq = dense2
```

On the left of the equals sign is the item than can be filled, and on the right side is the *Qt* brush to be used. The defaults for most are solid fill.

The entry `empty` isn't too useful; best to leave it set to 'solid'. The entry `note` affects the fill of normal/selected notes. The best values are either 'lineargradient' (the default) or 'solid'. The entry `scale` affects the fill for the piano roll scale. The hatching used here makes it easier to recognize that the scale is just there for orientation. The entry `backseq` affects the fill of the background sequence. The hatching used here helps further distinguish the real notes from the background notes.

16.2 Palettes Summary

There are some obvious enhancements to this scheme, including increasing the number of palette items, synchronizing the palette with the current desktop them semi-automatically, and providing a user interface to drag-and-drop colors.

16.3 Theming

When using a non-Qt-based window manager or desktop manager, such as our favorite, *Fluxbox*, in conjunction with *GTK+* themes, there can be issues on some *Linux* distros.

First, if using a Gtk theme setter (e.g. *gtk-chtheme* or *lxappearance*), one wants to use *qt5ct* to set Qt to work with Gtk themes. For this to work well, use this setting in the `.bashrc` or `.profile` file:

```
export QT_QPA_PLATFORMTHEME=gtk2
```

If not, use the following setting:

```
export QT_QPA_PLATFORMTHEME=qt5ct
```

Another option is to provide an executable script like the following, giving it a name such as `dseq66` (for a dark themed *Seq66*) to distinguish it from the normal-themed `qseq66`.

```
#!/bin/sh
# Use dark coloring on qseq66, as we have configured in qt5ct.
QT_QPA_PLATFORMTHEME=qt5ct qseq66
```

One might still encounter the issue that, with a Gtk theme, applications take about 20 to 30 seconds to start up!

Another issue is that some Qt themes might upset the sizing of button or text.

Also see the style-sheet discussion in section 12 "Seq66 Configuration" on page 134.

17 Seq66 Keyboard and Mouse Actions

This section presents some tables summarizing keyboard and mouse actions available in *Seq66*. The mute keys and group keys are also well described in the keyboard options for the main window (section 5.1.3 "Pattern Keys and Clicks" on page 68).

It does not cover the "fruity" mouse actions, as this mode of mouse-handling is not supported in *Seq66*.

This section describes the keystrokes that are currently hardwired in *Seq66*. This description only includes items not defined in the 'ctrl' file. That is, hardwired values. "KP" stands for "keypad". The effect that keystrokes have depends upon which window has the keyboard/mouse focus. For a possibly more up-to-date list, see the **Automation Defaults** and **Hardwired Keys** tabs in the `control_keys.ods` spreadsheet which is installed in the installed shared-data directory (e.g. `/usr/share/doc/seq66-0.99`).

17.1 Keyboard Control

Seq66 provides a plethora of keyboard controls for user-interface actions, note-modification, zooming, and pattern control. Most of these controls (not all) are easy to change by editing the appropriate 'ctrl' configuration file, stored in one of the following directories, depending on the operating system:

```
/home/username/.config/seq66/qseq66.ctrl
C:/Users/username/AppData/Local/seq66/qpseq66.ctrl
```

There are also some extended examples present in the *Seq66* `data/linux` and `data/samples` directory. Note that keyboard and MIDI control settings have been consolidated into a single table in

the 'ctrl' file. The `[mute-group]` control section has been moved to it's own 'mutes' file.

There are a number of "gotchas" to be aware of when assigning keys to the actions in the 'ctrl' file:

- Some of the keystrokes are hard-wired, such as "arrow" keys (for controlling play-lists), "page up/down" keys, the "zoom" keys, and some of the "Ctrl" keys.
- *Seq66* has appropriated the Shift key so that a a Shift-Left-click on a pattern slot opens up the corresponding set (based on pattern number) in an external live grid, and a Ctrl-Left-click requests a new pattern. For the group-learn feature, the `Shift` key is automatically enabled, using an "auto-shift" feature. Thus, using characters that require the Shift key while clicking, such as `{` and `}`, becomes surprising. Instead, look to the remaining keys: `F11`, `F12`, and the "keypad" keys if more keystrokes are wanted.

`[keyboard-control]`. We won't attempt to cover every key-control item, just the categories. Some items might be discussed in other parts of this manual. Remember that most keys have been consolidated with the MIDI controls. Also remember that the 'ctrl' file contains comments and an orderly layout to make it easier to understand and to edit.

An additional key definition is shown for the pause key. By default, the pause key is the period (`".."`), but that can be changed in the 'ctrl' file. In the piano rolls it is a hard-wired keystroke.

A goal of *Seq66* is being able to edit a pattern using mainly the computer keyboard. *Seq66* supports two pattern-slot modifier keys. The first modifier key causes the usual pattern-toggle key (hot-key) for a given slot to instead bring up the pattern editor. By default, this key is the equals (`"="`) key. The second modifier key causes the usual pattern-toggle key (hot-key) for a given slot to instead bring up the event editor. By default, this key is the minus (`"-"`) key. Both of these keys are configurable.

Some of the keys have positional mnemonic value. For example, for BPM control, the semicolon is at the left (down), and the apostrophe is at the right (up).

The **slot shift** key is useful when using pattern grids larger than 8 x 4 patterns. Pressing the slot-shift key basically adds 32 to the pattern number of the slot-key that is pressed. The default key is the forward slash (`"/"`) key.

A **snapshot** is a briefly-preserved state of the patterns. One can press (and release) the snapshot key, change the state of the patterns for live playback, and then press the snapshot key again to revert to the state when the snapshot key was first pressed. The default key is the `Ins` key.

To **queue** a pattern means to ready it for playback upon the next repeat of a pattern. A pattern can be armed immediately with a hot-key, or it can be queued to play back the next time the pattern repeats. A pattern can be queued by holding the queue key (defined in the 'ctrl' file) and pressing a pattern-slot hot-key. Instead of the pattern turning on immediately, it turns on at the next repeat of the pattern. The default key is the `"o"` key.

Keep queue allows the queue to be held without holding down the queue button the whole time. First, press the keep-queue key. Next, hitting any of the slot hot-keys, no matter how many, sets up the corresponding pattern slot to be queued. Also, in keep-queue mode, clicking on the pattern slot will queue the pattern. The keep-queue mode is disabled by hitting the "queue" key again (any currently active queues remain active until finished). The default key is the backslash key, `"\"` key.

There is also a "Q" button to toggle the keep-queue status.

Oneshot causes a slot to be queued for only a single playback. The default key is the pipe, "`|`" key. Currently buggy.

10. Sequence toggle keys. Each of these keys toggles the playing/muting of one of the 32 loop/pattern boxes. These keys are layed out logically on the keyboard by default, and can also be shown in each loop/pattern box. Please note that we often call them "shortcut keys" or "hot-keys" where the context makes it clear that they apply to the armed/unarmed state of a pattern.

11. Mute-group slots. There can be up to 32 mute-groups. When activated, a mute-group sets the muted/unmuted status of the current "playing set" to the pattern-muting statuses of the selected mute-group. Each of these keys operates on the mute-grouping of one of the 32 stored mute groups. These keys are layed out logically on the keyboard by default, and consist of `Shift` versions of the sequence-toggle (hot) keys. Note that a mute-group key will be memorized only when *Seq66* is in *group-learn* mode.

12. Learn. To define the group of patterns for one mute group, press and hold the configured Learn key (the "el", "`l`" key by default, the hard-wired `Ctrl-L` key, or the "**L**" button in the user-interface. Then pressing one of the mute group keys (*lower case* version) will save the currently-playing pattern slots into the corresponding mute group. The default mute group keys must be the shifted version of the key, but one should not press the `Shift` key for this key. *Seq66* will automatically assign the corresponding key with `Shift` activated.

Note that, once in learn-mode, there is no way to cancel learn-mode except by selecting a mute-group keystroke or toggling the **Learn** ("L") button. Also see section 15 "Seq66 Mutes Master" on page 195.

13. Disable. It is the inverse **apostrophe** key by default. This key is the *group off* key.

14. Enable. It is the **igrave** (back-tick) key by default. This key is the *group on* key. A number of additional functions have been added to *Seq66*, and keystrokes have been provided for those new functions.

Note the **Song/Live toggle** key. The *song mode* normally is in effect only when playback is started from the **Song Editor**. Now this mode can be used from any window, if enabled by pressing this key. There is also a button in the main window for this function, which shows the current state of this flag. Note that this flag is also stored in the 'rc' configuration file, as well as this hot-key value, which defaults to `F10`.

The *JACK mode* is set via the **Edit** / **Preferences** / **JACK** / **JACK Connect** or **JACK Disconnect** buttons. This keystroke will toggle between JACK connect and JACK disconnect. The **Song Editor** will also have a **JACK** button. The hot-key for this function defaults to `F2`.

The *menu mode* indicates if the main menu of the main window is accessible or not. It is disabled during playback so that more hot-keys can be used without triggering menu functions. It can also be disabled by the user; the default hot-key is `F3`. This feature is needed because the original *Seq24* had numerous conflicts between the menu key bindings and the default key bindings for the main window.

Follow JACK is a feature ported from *Seq32*. The default key is `F4`. It determines if *Seq66* follows

JACK transport.

Fast forward is a feature ported from *Seq32*. The default key is `F6`. While this key is held, the song pointer will fast-forward through the song. This feature does not have a corresponding button.

Rewind is a feature ported from *Seq32*. The default key is `F5`. While this key is held, the song pointer will rewind. This feature does not have a corresponding button.

Pointer position is a feature ported from *Seq32*. The default key is `F7`. When this key is pressed, the song pointer will move to the current position of the mouse, snapped. This feature does not have a corresponding button.

Toggle mutes toggles the mute status of every pattern on every screen-set. It corresponds to the **Edit / Toggle mute all tracks** or the **Song / Toggle All Tracks** menu entries. There is also a button in the main window for this function, which shows the current state of this flag. Note that this hot-key value is stored in the 'rc' configuration file, and defaults to `F8`.

Tap BPM allows the user to "tap" in time with some other music, and see the tap sequence translated into beats/minute (BPM). There is also a "0" button for this function. After 5 seconds, this feature resets automatically, so the user can try again if not satisfied. At least two taps are needed for the BPM to be registered.

17.2 Main Window

The main window keystrokes are all defined via the options dialog and "rc" configuration file, or are stock window-management keystrokes. The main window has a very complete setup for live control of the MIDI tune via keystrokes. These actions are not included in table 9 "Main Window Support" on page 205.

Table 9: Main Window Support

Action	**Normal**	**Double**	**Shift**	**Ctrl**
e	—	—	—	Open song editor
l (el)	—	—	—	Enter Learn mode
Left-click slot	Mute/Unmute	New/Edit	Toggle other slots	—
Right-click slot	Edit menu	—	Edit menu	Edit Menu

The new mouse features of this window for *Seq66*, as noted in section 5 "Patterns Panel" on page 57, are:

- *Shift-Left-click*: Over one pattern slot, this action toggles the mute/unmute (armed/unarmed) status of all other patterns (even the patterns in other, unseen sets).
- *Left-Double-click*: Over a pattern slot, this action quickly toggles the mute/unmute status, which is confusing. But it ultimately brings up the pattern editor (sequence editor) for that pattern.

17.3 Performance Editor Window

The "performance editor" window is also known as the "song editor" window. It's main sections are the "piano roll" (perfroll) and the "performance time" (perftime) sections, discussed in the following sections. Also, some keystrokes are handled by the frame of the window.

- `Ctrl-z`. Undo.
- `Ctrl-r`. Redo.

17.3.1 Performance Editor Piano Roll

Note that the keystrokes in this table (see table 10 "Performance Window Piano Roll" on page 206) require that the focus first be assigned to the piano roll by left-clicking in an empty area within it. Otherwise, another section of the performance editor might receive the keystroke.

Table 10: Performance Window Piano Roll

Action	**Normal**	**Double**	**Shift**	**Ctrl**
Space	Start playback	—	—	—
Esc	Stop playback	—	—	—
Period (.)	Pause playback	—	—	—
Del	Cut section	—	—	—
c key	—	—	—	Copy
p key	Paint mode	—	—	—
v key	—	—	—	Paste
x key	Escape paint	—	—	Cut
z key	Zoom out	—	—	Undo
0 key	Reset zoom	—	—	—
Z key	Zoom in	—	—	Undo
Left-arrow	Move earlier	—	—	—
Right-arrow	Move later	—	—	—
Left-click	Select section	—	—	—
Right-click	Paint mode	—	Paint mode	Paint mode
Scroll-up	Scroll up	—	Scroll Left	Scroll Up
Scroll-down	Scroll down	—	Scroll Right	Scroll Down

This section of the performance editor also handles the start, stop, and pause keys. These can be modified in the **Options** / **Keyboard** page. A "section" in the performance editor is actually a box that specifies a trigger for the pattern in that sequence/pattern slot. Note that the "toggle other slots" action occurs only if *shift-Left-clicked* in the "names" area of the performance editor. *Left-click* is used to select performance blocks if clicked within a block, or to deselect them if clicked in an empty area of the piano roll. Also note that all scrolling is done by the internal horizontal and vertical step increments. Some features of this window for *Seq66*, as noted in section 7 "Song Editor" on page 100, are explained here:

- *p*: Enters the paint mode, until right-click is pressed or until the "x" key is pressed.

- *x*: Exits the paint mode. Think of the made-up term "x-scape".
- *z*: Zooms out the performance view. It makes the view look smaller, so that more of the performance can be seen. Opening a second performance view is another way to see more of the performance.
- *0*: Resets the zoom to its normal value.
- *Z*: Zooms in the performance view, making the view larger, so that more details of the performance can be seen.
- *Left Arrow*: Moves the selected item to the left (earlier in time) in the performance layout.
- *Right Arrow*: Moves the selected item to the right (later in time) in the performance layout.
- Once selected (rendered in grey), a pattern section (trigger) can be moved by the mouse. To move it using the left or right arrow keys, the paint mode must be entered, but only via the "p" key.

17.3.2 Performance Editor Time Section

- `l`. Set to move L marker.
- `r`. Set to move R marker.
- `x`. Escape ("x-scape") the movement mode.
- `Left`. Move the selected marker left.
- `Right`. Move the selected marker right.

This section of the performance editor is also known as the "measure ruler" or the "bar indicator".

Table 11: Performance Editor Time Section

Action	**Normal**	**Double**	**Shift**	**Ctrl**
l	Move L [1]	—	—	—
r	Move R [1]	—	—	—
x	Escape Move	—	—	—
Left-Click	Set L [2]	—	—	—
Middle-Click	—	—	—	—
Right-Click	Set R [2]	—	—	—

1. Activates movement of this marker using the left and right arrow keys. Movement is in increments of the snap value. This mode is exited by pressing the 'x' key. Also see note [2].
2. Controlled in the pertime section.

The features of this window for *Seq66* are:

- *l*: Enters a mode where the left and right arrow keys move the L marker, until the "x" key is pressed.
- *r*: Enters a mode where the left and right arrow keys move the R marker, until the "x" key is pressed.
- *x*: Exits the marker-movement mode.

17.3.3 Performance Editor Names Section

Table 12: Performance Editor Names Section

Action	Normal	Double	Shift	Ctrl
Left-Click	Toggle track	—	Toggle other tracks	—
Middle-Click	—	—	—	—
Right-Click	New/Edit menu	—	—	—

17.4 Pattern Editor Piano Roll Keystrokes

The pattern/sequencer editor piano roll is a complex and powerful event editor; table 13 "Pattern Editor Piano Roll" on page 209, doesn't begin to cover its functionality. Here are the keystrokes handled by the main frame of the piano roll:

- `Delete`. Deletes (not cuts) the currently-selected notes.
- `Backspace`. Same as `Delete`.
- `Left Arrow`, `Right Arrow`, `Up Arrow`,and `Down Arrow`. Moves the selected notes by one semitone in pitch vertically, or one snap step horizontally.
- `Ctrl-Left Arrow` and `Ctrl-Right Arrow`. Absolute left/right movement by a snap step. To be explored.
- `z` and `Z`. Zoom out (smaller) and zoom in horizontally.
- `v` and `V`. Zoom out (smaller) and zoom in vertically.
- `0`. Reset horizontal or vertical zoom.
- `Ctrl-Home`. Scroll leftward to the beginning of the piano roll (time 0).
- `Ctrl-End`. Scroll rightward to the end of the piano roll (the full length of the pattern).
- `Ctrl-a`. Select all notes. The selected notes, events, and data values are drawn in orange (by default).
- `Ctrl-c`, `Ctrl-x`, `Ctrl-v`, and `Ctrl-z`. Copy, cut, paste, and undo. There is no redo key, but there are user-interface buttons for undo and redo.
- `Page Down`. Scroll downward.
- `Page Up`. Scroll upward.
- `c`. Repitch the selected note according to the loaded note-mapper, if any.
- `f`. Edge-fix. To be determined.
- `o`. Toggle recording for the pattern.
- `p`. Enter paint/drawing mode for notes.
- `q`. Quantize selected notes. Currently **broken**.
- `t`. Tightened (partial quantize) selected notes. Currently **broken**.
- `r`. Randomize selected notes. Currently **broken**.
- `x`. Exit paint/drawing mode.

17.4.1 Pattern Editor Piano Roll

Here are the keystrokes handled by the piano roll: These keystrokes require that the focus be set to the piano roll by clicking in it with the mouse.

- `Ctrl-Left`. Shrink selected notes.
- `Ctrl-Right`. Grow selected notes.
- `Delete`. Remove selected notes.
- `Backspade`. Remove selected notes.
- `Home. Set sequence to beginnging of sequence`. (Verify!)
- `Enter, Return`. Paste the selected notes at the current position.

And here is the table, which includes items not described above:

Table 13: Pattern Editor Piano Roll

Action	**Normal**	**Double**	**Shift**	**Ctrl**
Del	Delete Selected	—	—	—
c	—	—	—	Copy
o	Toggle record	—	—	—
p	Paint mode	—	—	—
v	—	—	—	Paste
x	Escape Paint	—	—	Cut
z	Zoom Out	—	Zoom In	Undo
0	Reset Zoom	—	—	—
Left-Arrow	Move Earlier [1]	—	—	—
Right-Arrow	Move Later [1]	—	—	—
Up-Arrow	Increase Pitch	—	—	—
Down-Arrow	Decrease Pitch	—	—	—
Left-Click	Deselect	—	—	—
Right-Click	Paint mode	—	Edit Menu	Edit/Edit Menu
Left-Middle-Click	Grow Selected	—	Stretch Sel.	—
Scroll-Up	Zoom Time In	—	Scroll Left	Zoom Time In
Scroll-Down	Zoom Time Out	—	Scroll Right	Zoom Time Out

1. Once selected (and thus rendered in grey), a pattern segment can be moved by the mouse. To move it using the left or right arrow keys, the paint mode must be entered, but only via the `p` key – the right mouse button deselects the greyed pattern. Too tricky, we might try fixing it later.

Features of this window section for *Seq66*, as noted in section 6.4.1 "Pattern Editor / Piano Roll Items" on page 86, are:

- *p*: Enters the paint mode, until right-click is pressed or until the `x` key is pressed. Notes are added by clicking or click-dragging.
- *x*: Exits ("x-scapes") the paint mode.
- *z*: Zooms out.
- *0*: Resets zoom to its normal value.
- *Z*: Zooms in.
- .: The period (configurable) does the pause function.
- *Left Arrow*: Moves selected events to the left.

- *Right Arrow*: Moves selected events to the right.
- *Up Arrow*: Moves selected notes upward in pitch.
- *Down Arrow*: Moves selected notes downward in pitch.

17.4.2 Pattern Editor Event Panel

- `Ctrl-x`. Cut.
- `Ctrl-c`. Copy.
- `Ctrl-v`. Paste.
- `Ctrl-z`. Undo.
- `Delete`. Delete (not cut!) the selected events.
- `p`. Enter "paint" (also known as "adding") mode.
- `x`. Escape ("x-scape") the paint mode.

17.4.3 Pattern Editor Data Panel

Currently, no keystroke support is provided in the data panel. One potential upgrade would be the ability to change the value of the event with the Up and Down arrow keys.

17.4.4 Pattern Editor Virtual Keyboard

Table 14: Pattern Editor Virtual Piano Keyboard

Action	**Normal**	**Double**	**Shift**	**Ctrl**
Left-Click	Play note	—	—	—
Right-Click	Toggle labels	—	—	—

17.5 Event Editor

- `Down`. Move one slot down.
- `Up`. Move one slot up.
- `Page Down`. Move one frame down.
- `Page Up`. Move one frame up.
- `Home`. Move to top frame.
- `End`. Move to bottom frame.
- `Asterisk, KP Multiply`. Delete the currently-selected event.

18 Seq66 In Windows

This section discusses installing and using the basics of *Seq66* in *Microsoft Windows*. Additional trouble-shooting information can be found in the installed file data directory.

```
C:/Program Files/Seq66/data/readme.windows
```

Another useful reference is *Working with MIDI on Windows* [41].

18.1 Windows / Seq66 Installation

For *Seq66*, the installer executable is now part of each "major" release (currently 0.99.6). There are tricks to getting the installer (e.g. `seq66_setup_0.99.6-x64.exe`) properly. One can't just right-click and save the link. Instead, click on the link. Then look for a "Download" button, and click that.

Installation itself is straightforward. Run the installer (e.g. `seq66_setup_0.99.6-x64.exe`). Accept the license terms (*GNU GPL 2 or 3*), make sure all components are selected, accept the default install directory, and click through until the installation is done.

(Note that there might also be a `qpseq66-release-package-0.99.6.7z` portable installer than can be downloaded. If not, request one! Just extract that file where desired. Note that the *Windows* version is a 64-bit application. We do not currently build a 32-bit version, due to our lack of a 32-bit Windows platform.)

Seq66 is installed at `C:/Program Files/Seq66/qpseq66.exe`. One might want to create a desktop shortcut or a shortcut button on the *Quick Launch* bar; one can also go to the "Start" menu and search for "qpseq66.exe" or "Seq66".)

Before we discuss usage, we have to talk a bit about a Windows issue.

18.2 Windows / MIDI Mapper

Windows has a **MIDI Mapper**. This is a subsystem that redirects MIDI to a specific device. Supposedly it was removed in Windows 8 and beyond, but it still exists in Windows 10. However, the alleged Control Panel applet for the MIDI Mapper no longer exists.

In the Windows XP era, MIDI was exposed in the OS, and it had a setup applet in the *Sound and Multimedia Control Panel*. Users could select a default MIDI Out device from a list of all installed MIDI devices. All programs outputting a MIDI data stream (and no selected a specific MIDI Out device) played their stream to that device.

The MIDI Mapper is not real device but a "pipe"; it receives MIDI on input and sends it to an user-configured MIDI Out device. It was bundled with Windows, installed as MIDI Out device 0, preconfigured to use the first available device, the built-in *Microsoft GS Wavetable Synth*. It is a low quality software wave synth, installed as MIDI Out device 1. So on Windows, programmers had 2 well-known devices.

This works for MIDI software without a configurable output device; they use MIDI Out 0, and the wavetable synth generates the sound. This chain worked well: default users had working MIDI synthesis out of the box. Then Microsoft removed it in Windows 7. But it resurfaced in Windows 10, but without the MIDI Control Panel applet.

It gets more interesting. Assume we have a LaunchPad Mini and a Korg nanoKEY2 plugged in. Internally, Windows see the following devices (based on a trek in the debugger):

- **0**. MIDI Mapper.
- **3**. GS Wavetable Synth.
- **4**. NanoKEY2.
- **5**. LaunchPad Mini.

Our "portmidi" implementation opens them in the same order, but numbers them sequentially:

- **0**. MIDI Mapper.
- **1**. GS Wavetable Synth.
- **2**. NanoKEY2.
- **3**. LaunchPad Mini.

The MIDI Mapper, if present, grabs the wavetable synth, which prevents *Seq66* from opening it. So the *Seq66* code detects that the MIDI Mapper (port 0) exists, and when opening the wavetable synth (port 1) fails, it marks that port as "unavailable" and "locked", in order to ghost it in the user-interface and ignore it for error reporting (otherwise every startup would bring up an annoying and meaningless error message).

As an aside, for MIDI input, Windows does no mapping, and there are only two input devices (given the two plugged in above).

- **0**. NanoKEY2.
- **1**. LaunchPad Mini.

There are applications that try to get around this issue, such as the *CoolSoft MIDIMapper* and *VirtualMIDISynth* [15] or *MIDI OX* [16]. See the `readme.windows` file for more information.

18.3 Windows / Virtual Ports

Virtual ports are software MIDI ports that can be created for input and output, as noted in section 4.2.1.1 "Menu / Edit / Preferences / MIDI Clock" on page 37and section 4.2.1.2 "Menu / Edit / Preferences / MIDI Input" on page 41. To provide virtual ports in *Windows*, special software needs to be installed. One useful applications is *LoopMIDI* (see [12]).

> Virtual loopback MIDI cable for Windows 7 up to Windows 10, 32 and 64 bit. This software can be used to create virtual loopback MIDI-ports to interconnect applications on Windows that want to open hardware-MIDI-ports for communication.

With this software, one can create a number of extra MIDI ports. The main difference from *Linux* here is that the `-manual-port` option is *not* used. The virtual ports are treated as real ports in *Windows*.

18.4 Windows / Seq66 Startup

Now run `C:/Program Files/Seq66/qpseq66.exe`. It might generate an error, though we believe we have suppressed this error:

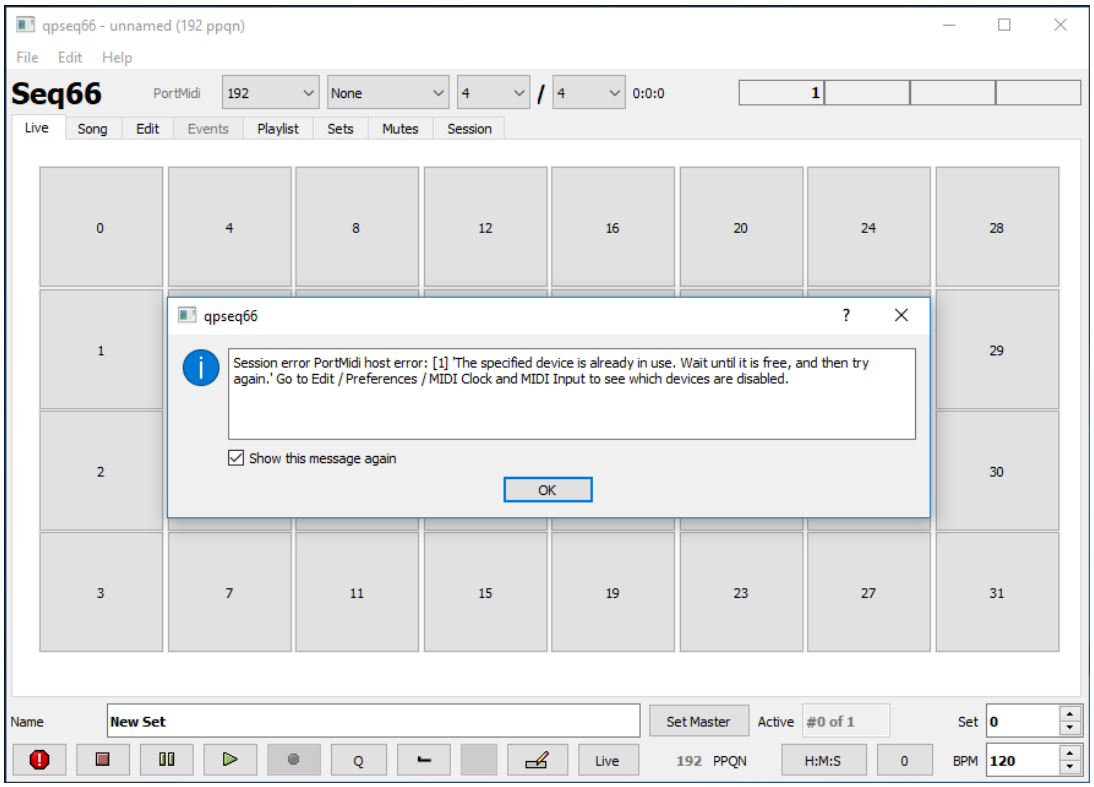


Figure 54: Seq66 First Startup in Windows

This error occurs on *Windows 10* because the 0th port, the *Microsoft MIDI Mapper*, grabs access to the 1st port, the *Microsoft GS Wavetable Synth*. Thus, the wavetable synth is "unavailable" and "locked". This message will appear at startup for "unavailable" ports, but not for a "locked" port; one can also just click **OK**, but startup messages can also be suppressed using an option in **Edit** / **Preferences** / **Display**.

Run the *qpseq66.exe* shortcut and load a tune from the directory `C:/Program Files/Seq66/data/midi`.

Navigate in the file explorer to `C:/Users/your_user_name/AppData/Local/seq66` and open `qpseq66.rc`, the main configuration file for *Seq66*. It will look like this:

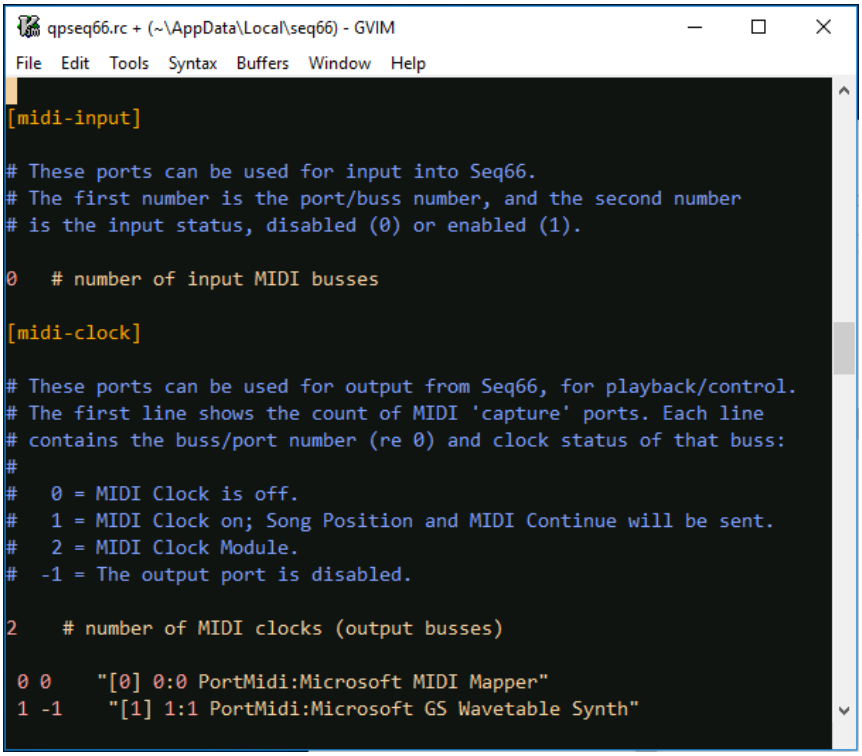


Figure 55: 'rc' File After Exiting First Startup

The `[midi-input]` section indicates there are no input ports in Windows when no MIDI device is connected to the computer. The `[midi-clock]` section indicates there are two output ports, and that port 1 is disabled. So one should be able to play a tune to the MIDI Mapper and hear it, if output is directed to port 0.

Go to **Edit** / **Preferences** / **MIDI Clock**. It will look like this:

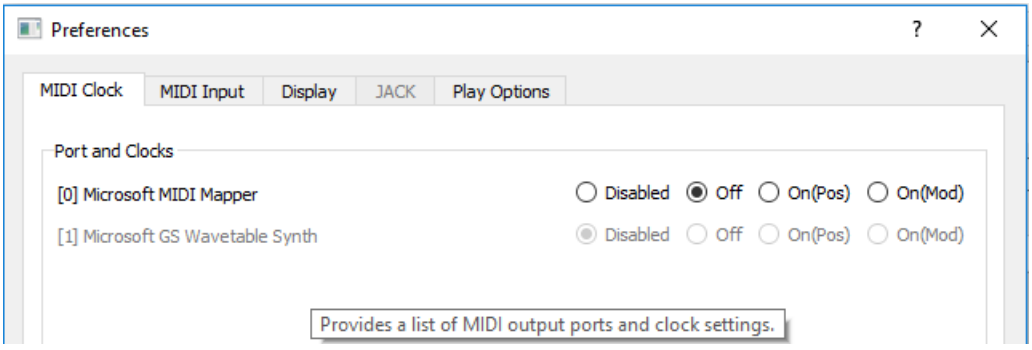


Figure 56: MIDI Output Settings at Second Startup

Next select **File** / **Open** and select this sample tune:

```
C:/Program Files/Seq66/data/midi/b4uacuse-gm-patchless.midi
```

(In the following figure, the " (x86)" is due to an early incorrect installation; *Seq66* on Windows is a 64-bit application only at this time.)

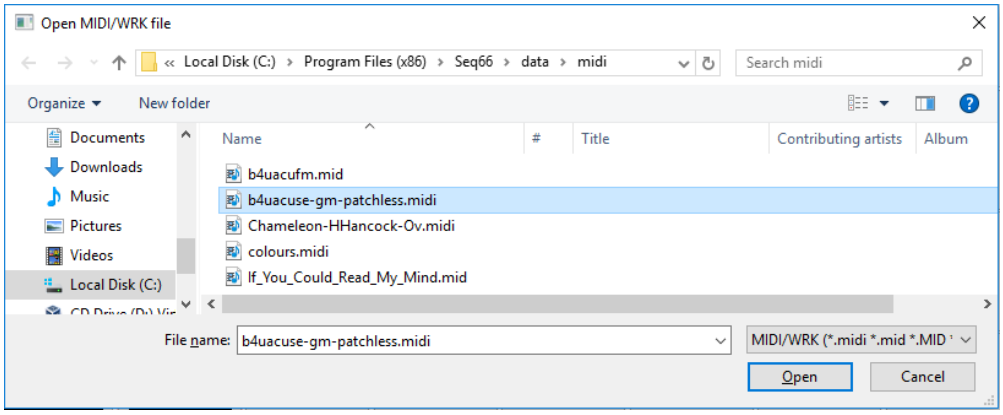


Figure 57: MIDI File Selection

After clicking **Open**, the following set of patterns is shown. Note the two highlighted areas, "Output Selector" and "Song/Live Button".

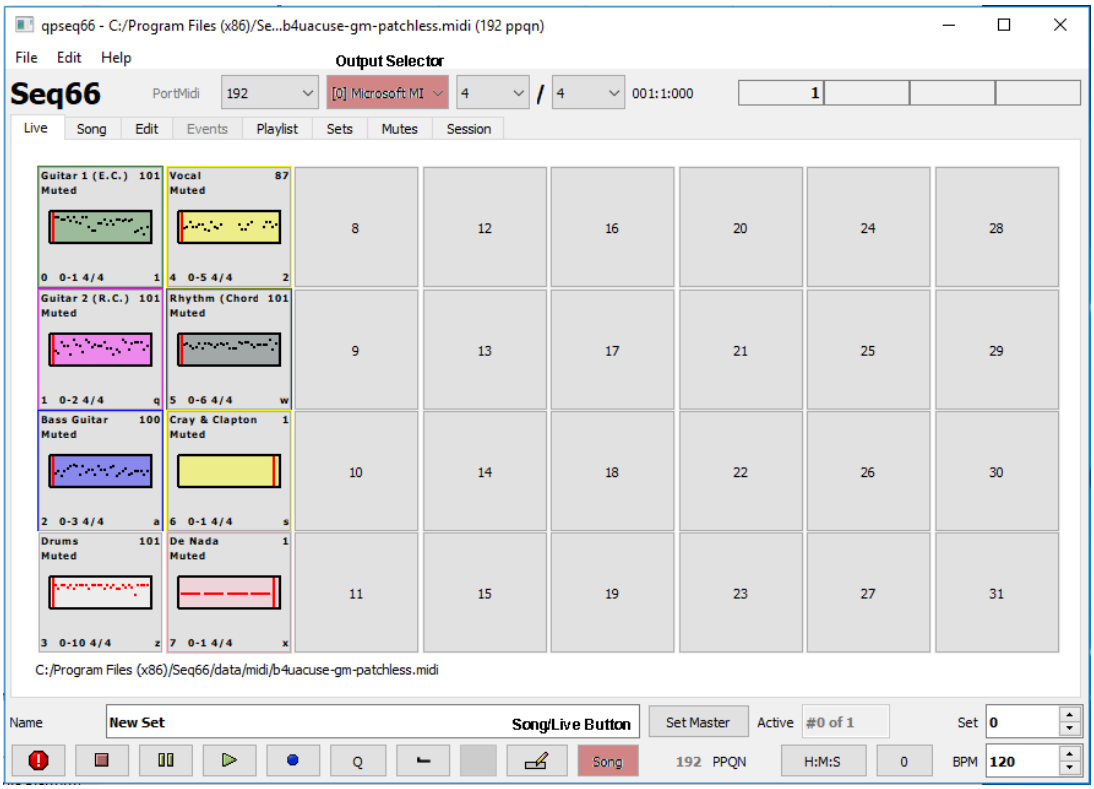


Figure 58: Opened MIDI File

At the top, select port 0 (the MIDI Mapper) from the "Output Selector". This *modifies* the MIDI file so that all MIDI output will go to port 0.

At the bottom, click the "Song/Live Button" until it reads "Song". This will access track layouts that turn on all of the patterns. These layouts can be seen by selecting the **Song** tab.

Now click the play button (green triangle). The song should play properly. (On our test Windows 10 setup in a virtual machine, playback is ragged, but fine on a normal Windows installation on hardware.)

Overall, the *Windows* version and the *Linux* version work essentially the same. The *Linux* version can use the *ALSA* and *JACK* MIDI engines, while the *Windows* version uses a refactored *PortMidi* engine that is part of the *Seq66* project.

The *PortMidi* engine should also work with *MacOSX*, but, since we don't have a Mac, we haven't been able to build and test on that platform. Again, for trouble-shooting, also see the installed text file:

```
C:/Program Files/Seq66/data/readme.windows
```

18.5 Windows / Keyboard Issues

The keystroke processing of *Seq66* is based on looking up *Qt* key scan codes and then finding the matching key modifiers (e.g. Shift or Ctrl) and native virtual key numbers. When a match is found, a *Seq66*-specific ordinal value ranging from 0 to 255 is found, and that is used to look up the command or function associated with that keystroke.

The native virtual key numbers are how the operating system encodes the various hardware keys on the keyboard. These numbers vary between *Linux*, *Windows*, and *Mac*. All these numbers are in a table, and there is (currently) a table for *Linux* and a table for *Windows*.

The table for *Windows* was recently completed, but it is likely that there are some native virtual key numbers that were not assigned correctly. Report any issues.

There is also a table that corrects for the French AZERTY layout. It is a good bet that this table also has errors. Report any issues.

19 ALSA

This section describes some details concerning the ALSA support of *Seq66*. Currently, it just includes some tricks that might be useful.

19.1 ALSA / Through Ports

When running the commands `aplaymidi -l` or `arecordmidi -l`, one sees something like the following. Notice that there is only one Thru port. The MIDI Through port is useful for...? They are virtualized hardware MIDI loopbacks that make it so programs that only output to "hardware" ports can use them to control and sequence other programs in an ALSA or JACK session. Also see how to increase the number of ALSA MIDI Thru ports in [2].

```
Port    Client name                     Port name
14:0    Midi Through                    Midi Through Port-0
28:0    nanoKEY2                        nanoKEY2 MIDI 1
 . . .
```

One minor issue with the system MIDI Thru port is that, if it and another input port (e.g. VMPK) are both enabled, then each note emitted by VMPK is doubled. Be aware.

> "ALSA always creates 1 MIDI through port. Since I work with Windows music applications via Wine, and because MIDI through ports are everything-proof, how can I increase the amount of MIDI through ports created by ALSA?"

To add more Thru ports, first use `modinfo` to see information about the kernel module `snd-seq-dummy`. Part of the output is shown here:

```
$ /sbin/modinfo snd-seq-dummy
filename:    /lib/modules/5.7.0-1-amd64/kernel/sound/core/seq/snd-seq-dummy.ko
alias:       snd-seq-client-14
description: ALSA sequencer MIDI-through client
author:      Takashi Iwai <tiwai@suse.de>
name:        snd_seq_dummy
parm:        ports:number of ports to be created (int)
parm:        duplex:create DUPLEX ports (bool)
```

Edit the following file (create it if necessary) to add a line to change the number of Thru ports. We use the '#' prompt to indicate root access or usage of `sudo`. Save it. No need to reboot, just remove and reinsert the module with the "mod" commands:

```
# vi /etc/modprobe.d/alsa-base.conf
options snd-seq-dummy ports=2
# rmmod snd_seq_dummy
# modprobe snd_seq_dummy
```

Then listing the ports will show:

```
Port    Client name                     Port name
14:0    Midi Through                    Midi Through Port-0
14:1    Midi Through                    Midi Through Port-1
28:0    nanoKEY2                        nanoKEY2 MIDI 1
 . . .
```

This will, of course, throw off the *Seq66* port numbering, unless one has implemented port-mapping (section 21 "Port Mapping" on page 227).

19.2 ALSA / Virtual MIDI Devices

The "manual" ports of *Seq66* are "virtual" ports. From *The Linux MIDI-HOWTO* [13]:

> MIDI sequencers like to output their notes to MIDI devices that normally route their events to the outside world, i.e., to hardware synths and samplers. With virtual MIDI devices one can keep the MIDI data inside the computer and let it control other software running on the same machine. This HOWTO describes all that is necessary to achieve this goal.

To use ALSA's virtual MIDI the `snd-card-virmidi` module must be present.

```
# Configure support for OSS /dev/sequencer and /dev/music (/dev/sequencer2)
# (Takashi Iwai: unnecessary to alias beyond the first card, i.e., card 0)
alias sound-service-0-1 snd-seq-oss
alias sound-service-0-8 snd-seq-oss
# Configure card 1 (second card) as a virtual MIDI card
sound-slot-1 snd-card-1
alias snd-card-1 snd-virmidi
```

More to come, such as an explanation of `aconnectgui`.... Also see the article about ALSA and JACK MIDI [14].

19.3 ALSA / Trouble-Shooting

This section describes some trouble-shooting that can be done with ALSA.

19.3.1 ALSA / Trouble-Shooting / ALSA Breakage

We have seen ALSA MIDI support break, perhaps due to a system update, on *Arch Linux*.. The first symptom is a prompt when *Seq66* starts:

```
Internal error. Creating master bus maps failed; check MIDI drivers
```

One can verify that it is not a *Seq66* issue by running `aplaymidi` or `arecordmidi`:

```
$ aplaymidi -l
ALSA lib seq_hw.c:529:(snd_seq_hw_open) open /dev/snd/seq failed: No
such device
Cannot open sequencer - No such device
```

If the command `modprobe snd_seq` run as `root` does not work, a reboot will.

19.3.2 ALSA / Trouble-Shooting / MIDI Clock

MIDI clock is a signal broadcast to ensure several MIDI-enabled synthesizers or sequencer stay in synchronization. Clock events are sent at a rate of 24 pulses per quarter note. THey maintain a synchronized tempo for synthesizers that have BPM-dependent voices, and for arpeggiator synchronization.

19.3.2.1 ALSA MIDI Clock Send

To verify that *Seq66* sends MIDI clock, the easiest way in ALSA is to run the following command and set *Seq66* to send MIDI Clock to the `aseqdump` port that appears after restarting *Seq66*:

```
$ aseqdump
Waiting for data at port 129:0. Press Ctrl+C to end.
Source  Event                  Ch  Data
  0:1   Port subscribed            128:0 -> 129:0
```

Once set up, start playback on *Seq66*. The result should be a never-ending stream of rapid MIDI clock messages.

19.3.2.2 ALSA MIDI Clock Receive

To verify that *Seq66* receives MIDI clock in ALSA, use a sequencer that can emit MIDI Clock in ALSA. The precursor to *Seq66*, *Seq24*, can be used, as well as *Seq66*. First, run one of the following commands:

```
$ seq24 -m
$ qseq66 --alsa --manual --verbose
```

This creates a bunch of virtual ALSA MIDI ports. In the **MIDI Clock** for either application, select the **On** option for the first port. Then run *a debug version of Seq66* in a terminal window with the following options:

```
$ qseq66 --alsa --auto-ports --verbose --client-name seq66debug
```

This sets up to auto-connect ALSA MIDI ports. In **Edit** / **Preferences** / **MIDI Input**, checkmark the first "seq24" port. There should be no need to restart. Then start playback in *Seq24*. *Seq66* should display a rapid stream of MIDI Clock messages. If not, check the setup and, if correct, report a bug!

19.3.2.3 VMPK Issues

On our *Ubuntu 18* system, the *VMPK* virtual keyboard (version to be determined) has issues:

```
Seq66:  input FIFO overrun
VMPK:   RtMidiOut::sendMessage: error sending MIDI message to port.
        RtMidiIn::alsaMidiHandler: unknown MIDI input error!
```

In addition, we find a massive spewage of events in the pattern. This does not occur on our *Ubuntu 20* system with *VMPK v. 0.7.2*. However, with that version, one must go to its **Edit / Preferences** dialog and make sure that MIDI I/O is enabled and that both input and output are set to ALSA.

Another issue occurred when we tried testing with multiple instances of VMPK. Both would appear in the MIDI Inputs list, with the same name but different client numbers. However, input would come in on the same buss number, no matter which VMPK instance was played. Thus, it's basically useless to run more than one VMPK instance.

See the *VMPK* website [40].

20 JACK

This section describes some details concerning the JACK support of *Seq66*. As with *Seq24*, *Seq66* has JACK transport support. JACK supposedly works with *Windows*, but we do not provide a JACK MIDI engine for that system at this time. The JACK support is very loosely based on the *RtMIDI* project [32]. This mode also supports fallback to ALSA if the JACK server is not running. *However*, if the `jack-dbus` is running, then a pair of virtual ports are created.

To enable the JACK transport support at run-time on *Linux*, the options `-j`/`--jack-transport` (deprecated, a legacy option), `-S`/`--jack-slave` (the new version of the transport option), `-J`/`--jack-master`, and `-C`/`--jack-master-cond` are available. `-g`/`--no-jack-transport` disables JACK transport even it JACK is running. Also see section 12.3.14 "'rc' File / JACK Transport" on page 150.

Note that the JACK transport options are *separate* from JACK MIDI. To enable JACK MIDI, the options `-t`, `--jack-midi`, and `-jack` are available. Note that the `-jack-midi` and `-jack` options both enable JACK MIDI.

If one wants to use *Seq66* and USB devices with JACK MIDI, one needs to expose the USB ports to JACK using the command `a2jmidid --export-hw`.

If `jackdbus` (Linux) is installed on the system, it is recommended to use `jack_control` or the wrapper script `jackctl` provided by *Seq66*. The following sections discuss the JACK transport support and the native JACK MIDI support.

20.1 JACK / Transport

JACK transport support is *separate* from native JACK MIDI support, and uses a separate JACK client. The JACK transport client is an invisible client with the name "seq66master" or "seq66slave", while the JACK MIDI client is visible in *QJackCtl*, and the ports created are part of the "seq66" client.

Seq66 can be configured to run without JACK transport, with JACK transport as a "slave" (i.e.

"client"), as JACK master, or as JACK master if there is no other master detected. Transport clients were formerly known as "transport slaves", and that term seems more clear than "client".

The *timebase master* continuously updates extended position information, counting beats, timecode, etc., while the transport is rolling. There is at most one master active at a time. If no timebase master, frame numbers are the only position information available. If a new client takes over, the former master timebase is no longer used. Taking over the timebase may be done conditionally, in which case the takeover fails when there is a master already. The existing master can release it voluntarily, if desired. There are some playback functions that JACK transport does not address:

- Variable speed playback.
- Reverse playback.
- Looping.

As per the rules of JACK, any client can start and stop the transport, and the other clients will follow suit. When *Seq66* is a JACK client, it will accept beats/minute (BPM) changes from another client that is running as master. When *Seq66* is master, changes to its BPM will be transmitted to the other clients.

Rules of JACK Transport:

- Any client can be master, no need for one master client.
- Multiple clients can change the transport state; any client can start/stop playback or seek to a new location.
- The transport timebase (tempo) is set and maintained by one of the clients.

Here are some test JACK Transport scenarios using *Seq66* and *Hydrogen* running under JACK. We currently have to restart Seq66 between each scenario.

Seq66 as JACK Slave, Hydrogen as JACK Slave

Either application can start and stop playback on both applications. The BPM (beats-per-minute) are changed independently on each application.

Seq66 as JACK Slave, Hydrogen as JACK Master

Either application can start and stop playback on both applications. Changing BPM on Hydrogen also changes it on Seq66, but not the other way around. Seq66 cannot change its BPM via the user-interface.

Seq66 as JACK Master, Hydrogen as JACK Slave

Either application can start and stop playback on both applications. Changing BPM on Seq66 also changes it on Hydrogen. If changed on Hydrogen, the BPM immediately returns to Seq66's BPM.

Seq66 as JACK Master, Hydrogen as JACK Master

If Hydrogen is already running, as soon as Seq66 starts, Hydrogren relinquishes JACK Master. If Hydrogen comes up after Seq66, it retains JACK Master status. Both applications can control start/stop and BPM

Note that, in *Seq66*, a changed BPM resets to the file-value after restart. Bug or feature?

20.2 JACK / Native MIDI

Currently, *Seq66* will connect to a JACK client automatically only at startup, where it will connect to all JACK clients that it finds. If it can't find a JACK client, then it will fail to register a JACK port, and cannot play.

The other option is to set up virtual ports using the `--manual-ports` or `--options virtual=o,i` options, and then to manually connected these ports to the desired MIDI devices or applications using *QJackCtl* (for example).

To run with JACK MIDI, just make sure JACK is running, then start *Seq66*, which will detect JACK and use it. If it instead opts to run with ALSA, edit the 'rc' file to set up `midi_jack`, or add the `-t`, `--jack-midi`, `--jack` or option to the command-line. If *Seq66* doesn't find JACK, it will fall back to ALSA.

The JACK (`-t`) and ALSA (`-A`) options are sticky options. That is, they are saved to the 'rc' configuration file at exit, so one does not have to specify them in subsequent *seq66* sessions.

20.2.1 JACK / MIDI Output

By default (or depending on the 'rc' configuration file), *Seq66* will automatically connect the ports that it finds to `seq66`. The following figure shows connections to a number of USB MIDI devices (purple) that have been bridged to JACK (red) by the `a2jmidid` daemon.

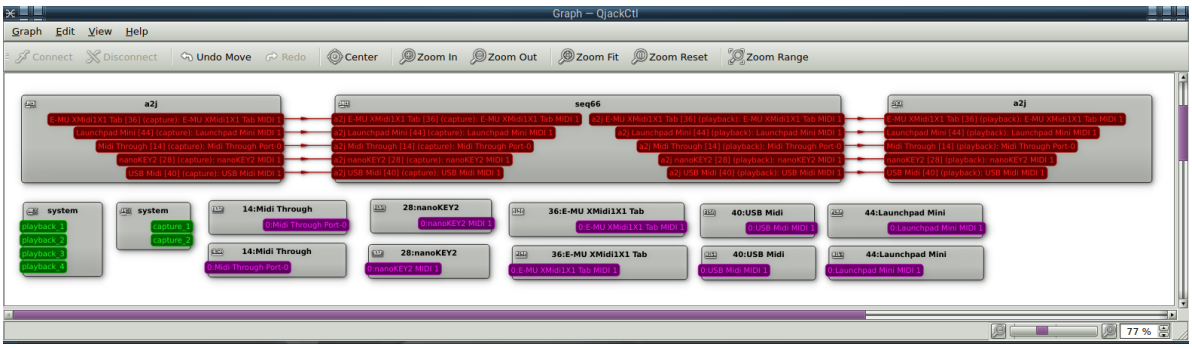


Figure 59: JACK MIDI Ports and Auto-Connect

Note that the ports in *Seq66* are named after the devices to which they are connected. The output ports available are shown in *seq66*'s **Edit** / **Preferences** / **MIDI Clock** tab. If USB devices are not shown, that means that the `a2jmidid` is not running. There is a `bash` script, `data/linux/startjack` that will run `jack_control` and `a2j_control` to start JACK and the `a2jmidid` daemon to provide full support. On our current setup, it creates devices with long names (abbreviated inside *Seq66*):

```
6    # number of MIDI clocks (output busses)
 0 0 "[0] 0:0 seq66:a2j Midi Through [14] (playback): Midi Through Port-0"
 1 0 "[1] 0:1 seq66:a2j Launchpad Mini [28] (playback): Launchpad Mini MIDI 1"
```

```
2 0 "[2] 0:2 seq66:a2j E-MU XMidi1X1 Tab [32] (playback): E-MU ... Tab MIDI 1"
3 0 "[3] 0:3 seq66:a2j nanoKEY2 [36] (playback): nanoKEY2 MIDI 1"
4 0 "[4] 0:4 seq66:a2j USB Midi [40] (playback): USB Midi MIDI 1"
5 0 "[5] 1:5 seq66:yoshimi-01 midi in"
```

A feature new with version 0.98.5 is that an output port can be enabled or disabled in the **Edit** / **Preferences** / **MIDI Clock** tab, and the change in status takes effect immediately. One minor issue is that, upon reenabling, a stray note or two might be emiited.

20.2.2 JACK / MIDI Input

The input ports also end up with long names:

```
5   # number of input MIDI busses
0 1 "[0] 0:0 seq66:a2j Midi Through [14] (capture): Midi Through Port-0"
1 1 "[1] 0:1 seq66:a2j Launchpad Mini [28] (capture): Launchpad Mini MIDI 1"
2 1 "[2] 0:2 seq66:a2j E-MU XMidi1X1 Tab [32] (capture): E-MU ... Tab MIDI 1"
3 0 "[3] 0:3 a2j:nanoKEY2 [36] (capture): nanoKEY2 MIDI 1"
4 0 "[4] 0:4 a2j:USB Midi [40] (capture): USB Midi MIDI 1"
```

When the check-box for a buss is selected, that input can be captured by `seq66`. An output port can be enabled or disabled in the **Edit** / **Preferences** / **MIDI Input** tab.

20.2.3 JACK / MIDI Virtual Ports

The manual-versus-normal port support for JACK MIDI is essentially the same as that for ALSA. The `--manual-ports` and `--options virtual=o,i` options provide "virtual ports". These are ports that do not represent hardware, but are created by applications to allow them to connect to other applications or MIDI devices.

The difference between manual/virtual ports and normal ports is that, while normal ports are automatically connected to the remote ports that exist in the system, the manual/virtual ports are just created, and one must manually connect them via, for example, the *QJackCtl* connections dialog.

So, if one wants *seq66* to automatically connect to all existing JACK MIDI ports, *do not* use the `-m`/`--manual-ports` option... use the `-a`/`--auto-ports` option. Both options apply to both ALSA and JACK. The **MIDI Clock** and **MIDI Input** tabs reflect what is seen in *QJackCtl*.

20.2.4 JACK / MIDI and a2jmidid

One thing we saw is that `seq66` can deal with the odd naming of JACK ports created by the *a2jmidid* application. One can see in the input and output lists shown earlier that that the `a2j` client creates entries for "Midi Through", software clients, and bridged USB MIDI devices. Again, if these automatic connections get in the way, run `seq66` in manual/virtual mode, or disable JACK auto-

connect in the `[jack-transport] jack-auto-connect]` setting (**Edit / Preferences / JACK / JACK auto-connect**.

Note that `a2jmidid -export-hardware` is roughly equivalentto JACK started with the `-Xseq` option.

To set up *JACK*, one can use the script shipped with *Seq66*, `data/linus/startjack`. It has the following requirements and dependencies:

- `qjackctl`. Provides a way to show the connections. It also can start *JACK*, but we use `jack_control` for that in this script.
- `jack_control`. Provides a way to start *JACK* and set up a number of *JACK* parameters. Part of the *Debian* `jack2d` package.
- `a2j_control`. Provides a way to configure and start the *ALSA*-to-*JACK* bridge to create bridges for all the hardware MIDI ports on the computer. Part of the *Debian* `a2jmidid` package.
- `yoshimi`. Provides a software synthesizer for MIDI playback.
- `yoshimi-b4uacuse-gm.state`. Provides a "General MIDI" setup for *yoshimi*. Located in the `data/linux` directory.
- *Editing*. One must edit the script to change the value of `HWPORT`

One can also edit the script to use another software synthesizer. Once ready, Run `startjack` and wait patiently for it to set up.

20.3 JACK / Trouble-Shooting

This section describes some trouble-shooting that can be done with JACK.

20.3.1 JACK / Trouble-Shooting / MIDI Clock

There are three common timecode formats: LTC (Linear Time Code), MTC (MIDI Time Code), and MIDI Clock, as well as JACK transport. This section describes MIDI Clock. It is an alternative to using JACK transport to synchronize devices.

MIDI Clock is not a timecode format but tempo-based time. The absolute reference point is expressed as beats-per-minute and bar:beat:tick. There is no concept of sample-locking for MIDI clock signals.

20.3.1.1 JACK MIDI Clock Send

To verify that *Seq66* is sending MIDI clock, the easiest way (using JACK) is to start the following command and set *Seq66* to send MIDI Clock to the `jack_mclk_dump` port that appears after restarting *Seq66*:

```
$ jack_mclk_dump
```

Once set up, start playback on *Seq66*. The result should be a 0xfa (MIDI Start) message, a never-ending stream of rapid MIDI clock messages, and then, upon stopping playback, a 0xfc (MIDI Stop) message.

Do not try to use *gmidimonitor...* while it displays Start and Stop, it does not display MIDI Clock.

20.3.1.2 JACK MIDI Clock Receive

To verify that *Seq66* receives MIDI clock in JACK, use a program that can emit MIDI Clock in JACK. We've tried `jack_midi_clock` to generate time code, but it didn't seem to work, even when connected to the application `jack_mclk_dump` used in the previous section. However, since we verified that *Seq66* can send MIDI Clock, we can use it to test itself.

First, run the following command using *debug version of Seq66*, so that we can see if it is receiving:

```
$ qseq66 --jack-midi --manual --verbose --client-name seq66debug
```

This creates virtual JACK ports that show up in a JACK patchbay as part of an application called "seq66debug". This lets us distinguish it from the regular version. Run the following *Seq66* command:

```
$ qseq66 --jack-midi --manual --verbose
```

Set MIDI Clock for "midi out 0" to "On" so that it will emit clocking. Restart `qseq66`. Then connect the "midi out" of "seq66" to the "midi in" of "seq66debug" in the JACK patchbay. Then start playback. *seq66debug* should display a rapid stream of MIDI Clock messages. If not, check the setup and, if correct, report a bug!

Note that the `-jack-midi` and `-jack` options both enable JACK MIDI.

20.3.2 JACK / Trouble-Shooting / JACK Ports

As noted in section 20.2.1 "JACK / MIDI Output" on page 222, and section 20.2.2 "JACK / MIDI Input" on page 223, `a2jmidid --export-hw` can be used to bridge USB MIDI hardware to JACK. However, on some systems, this bridge can be avoided, at the expense of a little confusion. For example, look at this partial output of the `jack_lsp` command:

```
system:midi_playback_1
system:midi_capture_2
system:midi_playback_2
system:midi_capture_3
system:midi_playback_3
system:midi_capture_4
system:midi_playback_4
fluidsynth-midi:midi_00
```

The "system" ports correspond to real MIDI devices on this system, but which ones? To find out, run `jack_lsp -aliases`. Here is just one device output:

```
system:midi_capture_2
   alsa_pcm:Launchpad-Mini/midi_playback_1
   Launchpad-Mini:midi/playback_1
system:midi_playback_2
   alsa_pcm:Launchpad-Mini/midi_capture_1
   Launchpad-Mini:midi/capture_1
```

So system MIDI device 2 is the *Launchpad Mini*, and will show up in *Seq66* as an I/O device on port 1. (Port 0 is usually the system MIDI Thru port). Of course, with this arrangement, port-mapping cannot be used. In the future, we may be able to ensure the aliases are always present in *Seq66*.

20.4 JACK / QJackCtl

This section is a placeholder for discussion of *QJackCtl*, its usage of the (deprecated) *JACK Session API*, and its "patchbay" (see [25]). Its a good idea to put "jackd -S" instead of just "jackd" for the server path. Running JACK in synchronous mode creates less Xruns in JACK2, which is now the default.

20.5 JACK / PulseAudio

After years of avoiding *PulseAudio* assiduously, we found a circumstance where we needed it. In using *OBS Studio* to make demonstration videos of *Seq66*, we struggled trying to get it to work with either *ALSA* or *JACK* directly. Setting up *PulseAudio* ended up being fairly easy on our *Debian Sid* system, not too cumbersome or intrusive (and also made it easier to use *Bluetooth* earbuds via *pulseaudio-module-bluetooth*). However, when we decided to a system reinstall on our main development laptop, we ended up installing *Ubuntu 20*, and it became a real pain.

This section discusses using *JACK* with *PulseAudio*. After installing *PulseAudio* and getting it to work with one's system (e.g. working with *Seq66*, *mpd*, *smplayer* etc. using *ALSA*, the next step is to install *JACK* support. In *Debian*-based systems, install *pulseaudio-module-jack*, *pulseaudio-utils*, and *pavucontrol*.

Then, assuming *qjackctl* is installed (which is useful for using *JACK* session management), then go to its menu **Setup...** / **Options** / **Execute script ...** and set up each of the "jack" scripts found in `data/linux/jack/pulseaudio`:

- **Execute script on Startup**: `/usr/local/bin/jack-pre-start.sh`
- **Execute script after Startup**: `/usr/local/bin/jack-post-start.sh`
- **Execute script on Shutdown**: `/usr/local/bin/jack-pre-stop.sh`
- **Execute script after Shutdown**: `/usr/local/bin/jack-post-stop.sh`

Other executable locations can be used if desired. These steps configure *qjackctl* to run those commands at the appropriate time. *PulseAudio* will recognize (via D-Bus) that *JACK* started, and

will route audio to it. When *JACK* is stopped, *PulseAudio* reverts to normal routing. Other *qjackctl* options from the **Misc** tab:

- **Start JACK audio server on application startup**: Check-mark it, unless one prefers to use the **Stop** button.
- **Stop JACK audio server on application exit**: Check if available, as desired.
- **Single application instance**: Check-mark it.
- **Enable ALSA Sequencer support**: Recommended.
- **Enable D-Bus interface**: Can leave unchecked.
- **Enable JACK D-Bus interface**: Can leave unchecked.
- **Save JACK audio server configuration to**: Leave unchecked. If the `.jackdrc` file is found, then *QSynth* might try to start JACK itself.

The following scenario describes what we are seeing on *Ubuntu 20.04.3*; the main anomaly is that playback of an application through *PulseAudio* doesn't resume on its own. Let's assume we are running *mpd* and it is playing tunes via *PulseAudio*. Then we start *QJackctl* configured as above. This action will silence *mpd*, but starting playback again will work. One can then run *Seq66* using JACK MIDI and JACK transport. Once finish, use *QJackctl* to stop JACK. This will silence *mpd* again. Then we have to restart *PulseAudio* (e.g. via the `repulse` script supplied in `data/linux/jack/pulseaudio`).

21 Port Mapping

Seq66, like *Seq24*, bases its I/O port scheme on buss/port *numbers*. This numbering scheme applies whether *ALSA*, *JACK*, or *Windows Multimedia* are used as the MIDI engine, and whether *Seq66* is running with "automatic" ports or "manual" (virtual, software-created) ports. These buss numbers range from 0 on upward based on the number of I/O MIDI ports active in the system. In "automatic" (non-virtual, non-manual) mode these ports represent the hardware MIDI ports and application MIDI ports. In "manual" mode, these ports represent virtual MIDI ports that the user can connect manually (on Linux).

Note: Port-mapping is now automatic; default MIDI I/O port-maps are created at first start, and are always active if virtual ports are not in use. Once created, one can edit the 'rc' file to rearrange the mapping as desired. In addition, issues with new or unavailable ports are noted in warning dialogs. (The startup warnings can be suppressed; see section 4.2.1.5 "Menu / Edit / Preferences / Display" on page 44.) The user can click the **Remap and restart** button, fix the loaded pattern to use existing ports, **OK** to ignore the warning,, or exit, determine the existing ports using `aplaymidi`, `arecordmidi`, or `jack_lsp`, and edit the maps directly in the 'rc' file.

The output bus or port for a given pattern can be determined by looking at the grid slots or by dumping a summary of the song to a text file. See section 4.3.3 "Menu / Help / Song Summary File.." on page 55. Also check the specified 'ctrl' file to see if it is using non-existent MIDI I/O ports.

A pattern/loop/sequence is assigned to output to a given port via a buss number saved *in the pattern*, in the tune. When a tune is loaded, each pattern outputs to the port number specified in the pattern.

A problem is that MIDI device setups can change, with devices being reordered, removed, or added to the MIDI devices available on the system. Or if the song is opened on someone else's computer. We do not want to store port names in the MIDI file. They can be too long, but, more importantly, they will differ between the systems of each user. They can even differ when switching from ALSA to JACK, or even versions of these MIDI engines. Better to let the user determine the port-mapping. Mapping allows the buss number stored with a pattern to be remapped to another buss number based on the "nick-name" (or JACK alias) of the port. It uses a simple lookup to map names to numbers.

The "nick-name" is a shortened version of the MIDI device name assigned by the system. For example, the long name of a MIDI port might be `[5] 44:0 E-MU XMidi1X1 Tab MIDI 1`. The nick-name is `E-MU XMidi1X1 Tab MIDI 1`. In order find the correct port number, the long name is checked to see if it *contains* the nick-name, and, if so, the corresponding port number is returned. The user can edit the 'rc' file to shorten the nick-name, if desired; a nick-name `E-MU XMidi1X1` would work.

In addition, under recent versions of *JACK*, there is a facility to get the "alias" of USB MIDI ports, even if the *a2jmidid* process is not running. This allows the user to see that the port name `system:midi_capture_5` is actually a "nanoKEY2" device.

So, with port-mapping enabled, one can set up the tune to record and play MIDI using the mapped ports, and later move to another computer, modify the port-maps int the 'rc' file to match, and record and play without issue.

The easiest way to start port-mapping (which is now automatic) is to go to **Edit** / **Preferences** / **MIDI Clock**. Here, we see the long MIDI port names made up by the system, along with the *JACK* aliases that can be retrieved, and which make it easy to see which devices are in play.

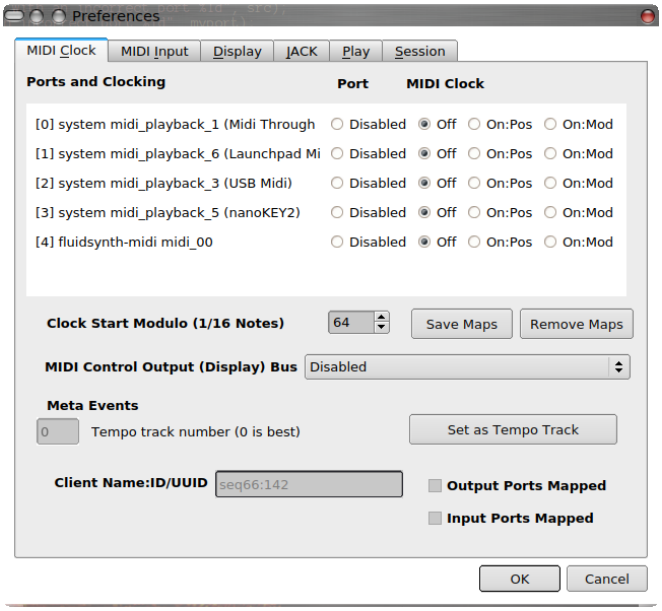


Figure 60: Clocks List Without Port Mapping

Note the `system:midi_playback` that is part of each port name. (There is a similar "capture"

portion for input ports). Click on the **Make Maps** button. This creates the initial I/O maps internally. Then either restart *Seq66* or go to the **Restart Seq66** button; this will save the new version of the 'rc' file.. Back in **Edit** / **Preferences** / **MIDI Clock**, one sees that the remapped names are in use.

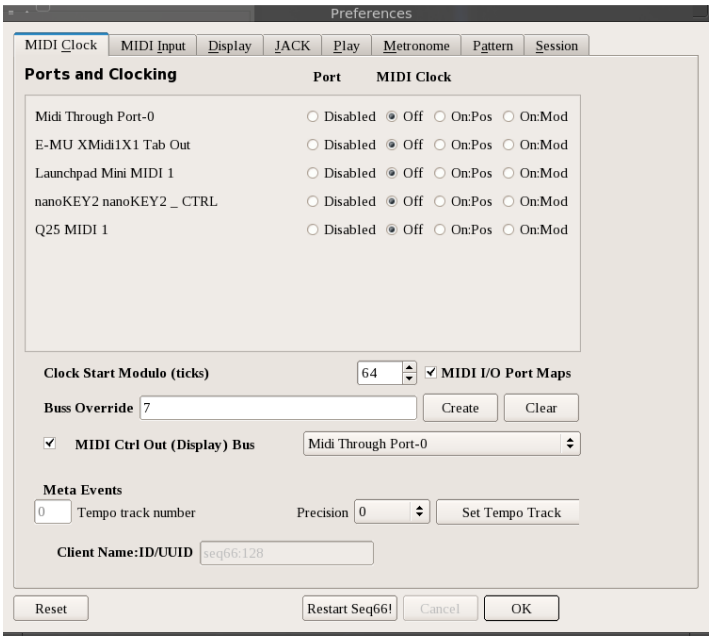


Figure 61: Clocks List After Port Mapping

Note that the short names generated are partial names of either the ALSA port name or are comprised of the JACK alias name for each port.

At startup, *Seq66* matches the port-map to the ports that exist on the system. If there is no matching system port for a mapped port, then that mapped port will show up as disabled in the port lists, and a warning should appear. If it makes sense, click on the **Remap and restart** button. Or later, go to **Edit** / **Preferences** / **MIDI Clock**. There, to remove the port mappings, click the **Clear Maps** button. To reconstruct the current setup, click on the **Make Maps** button to get the new mapping, and restart manually. As with the normal port listings, the port-mappings are saved and managed in the *Seq66* 'rc' file. One can also edit that file in a text editor to rearrange the mapped ports.

Note that one can also deactivate port-mapping. Both input and output will always be in the same state (activated or deactivated).

21.1 Input Port Mapping

The input ports are handling somewhat similarly. Here's another initial system input setup:

```
4      # number of input MIDI busses
0  1   "[0] 0:0 seq66:system midi_capture_1"     # 'Midi Through'
1  1   "[1] 0:1 seq66:system midi_capture_6"     # 'Launchpad Mini'
2  1   "[2] 0:2 seq66:system midi_capture_3"     # 'USB Midi'
```

```
 3  1   "[3] 0:3 seq66:system midi_capture_5"     # 'nanoKEY2'
```

To see an example from ALSA, look at the sample file `data/linux/qseq66.rc`. ALSA does not support aliases. (Neither do early versions of JACK.) Note that the "system:announce" ALSA buss is always disabled, as *Seq66* does not use it. In a future version of *Seq66* it may be removed, as it means ports have to be renumbered slightly when switching to JACK.. Here is the stored input port-map:

```
[midi-input-map]
 1      # map is active
 0  1   "Midi Through"
 1  1   "Launchpad Mini"
 2  1   "USB Midi"
 3  1   "nanoKEY2"
```

Other than being input devices, this input map works like the output (clocks) map. In the user interface dropdowns for input buss, if a map is active, it is put into the dropdown; any missing items are noted and are shown as disabled. If the map is not active, then only the actual system input ports are shown.

21.2 Output Port Mapping

Assume that the system has the following set of ports. These busses are stored in the 'rc' file when *Seq66* exits. Note the *JACK* aliases shown at the right as comments.

```
[midi-clock]
 5      # number of MIDI clocks (output busses)
 0  0   "[0] 0:0 seq66:system midi_playback_1"    # 'Midi Through'
 1  0   "[1] 0:1 seq66:system midi_playback_6"    # 'Launchpad Mini'
 2  0   "[2] 0:2 seq66:system midi_playback_3"    # 'USB Midi'
 3  0   "[3] 0:3 seq66:system midi_playback_5"    # 'nanoKEY2'
 4  0   "[4] 1:4 seq66:fluidsynth-midi midi_00"
```

To see an example from ALSA, look at the sample file `data/linux/qseq66.rc`.

If some items are unplugged, then this list will change, so we save it while still running *Seq66*: click the **Make Maps** button in the **Edit / Preferences/ MIDI Clock** dialog. The result is are new sections in the 'rc' file (one for clocks, one for inputs). Here is the clock map:

```
[midi-clock-map]
 1      # map is active
 0  0   "Midi Through"
 1  0   "Launchpad Mini"
 2  0   "USB Midi"
 3  0   "nanoKEY2"
 4  0   "fluidsynth-midi midi_00"
```

It is simpler, showing the nick-names (or aliases) of the ports. These index numbers can be used as buss numbers: they can be stored in a pattern, and used to direct output to a specific device.

If a pattern has stored a missing item as its output buss number, this number will not be found in the system list, so that the pattern will need to be remapped to an existing port.

Note that the mapping can be disabled by setting the first value to 0. In that case, *Seq66* uses buss numbers in the normal way. In the user interface dropdowns for output buss, if a map is active, it is put into the dropdown; any missing items are noted and are shown as disabled.

If the map is not active, then only the actual system output ports are shown in the user interface.

21.3 Port Mapping Example

This example shows that MIDI control and MIDI status displays work with port mapping. First, we run *Seq66*, save the ports for remapping, and exit the application. Looking in the 'rc' file, we tweak the maps:

```
[midi-input-map]
1       # map is active
0  0    "announce"
1  0    "Midi Through Port-0"
2  0    "Launchpad Mini MIDI 1"
3  0    "nanoKEY2 MIDI 1"
4  0    "Q25 MIDI 1"
5  0    "E-MU XMidi1X1 Tab MIDI 1"
```

And:

```
[midi-clock-map]
1       # map is active
0  0    "Midi Through Port-0"
1  0    "Launchpad Mini MIDI 1"
2  0    "nanoKEY2 MIDI 1"
3  0    "Q25 MIDI 1"
4  0    "E-MU XMidi1X1 Tab MIDI 1"
```

These two maps reflect the configuration at the time they were saved. They reflect the output of `arecordmidi --list` and `aplaymidi --list`. After unplugging and replugging some devices, we see that the *Launchpad Mini* has moved:

```
6   # number of input MIDI busses
3 0    "[3] 36:0 Launchpad Mini MIDI 1"

5   # number of MIDI clocks (output busses)
2 0    "[2] 36:0 Launchpad Mini MIDI 1"
```

On input, it has moved from buss 2 to buss 3. On output it has moved from buss 1 to buss 2. This can be verified by running *Seq66*, immediately exiting, and checking the `qseq66.rc` file. We edit

that file to add:

```
[midi-control-file]
"qseq66-lp-mini-alt.ctrl"
```

With the I/O maps shown above active in the 'rc' file, we can go to the 'ctrl' file (`qseq66-lp-mini-alt.ctrl`, available in the `data/linux` install/source directory) and set the following:

```
[midi-control-settings]
control-buss = 2              # maps to system input buss 3
midi-enabled = true

[midi-control-out-settings]
output-buss = 1               # maps to system output buss 2
midi-enabled = true
```

With this setup, the lights on the Mini light-up at start-up, and the buttons control the pattern, mute-groups, and automation features set up in the above-mentioned 'ctrl' file. The buss number will be replaced with the name of the device, e.g. `output-buss = "Launchpad Mini"`, if port-mapping is active.

Perhaps tricky, but once one has set up a whole suite of I/O device maps, one can reliably use these mapped port numbers to look up the actual system port numbers. For example, with the above setup, *Seq66* can be assured that output buss 1 will always go to the *Launchpad Mini*.

21.4 Port Setting SeqSpec

In the MIDI specification, there are two obsolete MIDI Meta events, "MIDI Channel" (0x20) and "MIDI Port" (0x21). These events were never endorsed by the MIDI Manufacturers Associated, but some versions of *Cakewalk* used them. In any case, one does not generally change the channel and port during playback.

However, *Seq66* needs to set the channel and port in order to determined where events in a pattern are output. These are specified by sequencer-specific (SeqSpec) events stored with each pattern.

If the patterns uses "Free" setting of channel (0x80), each channel event goes to whatever channel (0 through 15 internally) is specified in the event. If a channel is specified, then all channel events are redirected to that channel. See section 25.2.3 "SeqSpec c_midichannel" on page 254.

The port setting for each pattern is a number, as described earlier in this chapter. The output of a pattern goes to this numbered port in the midi-clock lists. Every pattern must have a port number. For imported tracks, this number defaults to 0. See section 25.2.1 "SeqSpec c_midibus" on page 252.

22 Seq66 Headless Version

Seq66 can be built as a command-line application. That is, *Seq66* can be run from the command-line, with no visible user interface. See the `INSTALL` file provided with the source code distribution. Note the `-both` bootstrap option to build the GUI and CLI versions of *Seq66* at the same time.

It can also be instantiated as a Linux daemon, for totally headless usage. Because there is no visibility into a headless process, the setup for `seq66cli` is tricky, and the musician must get used to blind MIDI control.

Also see section 12.2 "Command Line" on page 137

22.1 Seq66 Headless Setup

The first step in setting up a headless `seq66cli` session is to make sure that the GUI version (`seq66`) works as expected. The GUI and headless configurations need to do the following:

1. Access the correct inputs, especially a keyboard or pad controller that can be used for controlling the sequencer via MIDI, as well as inputing notes.
2. The desired MIDI controller input buss must be *enabled* by setting the active bit to "1" for that device.
3. The MIDI controller input must be configured with some `[automation-control]` values, so that the headless sequencer can stop and start playback, select the next playlist or song, or activate other sequencer controls. This is done by providing the name of a suitable `[midi-control-file]` ('ctrl') specified in the 'rc' file, and making sure it is marked as `active = true`.
4. Configure a separate MIDI buss to use to record from another MIDI instrument.
5. Access the desired outputs, in order to play sounds. This can sometimes be tricky, because *Seq66* can route all patterns to the same output, or can let the patterns decide the outputs for themselves.
6. The desired output busses must be *enabled* by setting the active bit to "1" for those device. Note that some MIDI controllers can be used to show the status of the music, and would also be configured in the 'ctrl' file.
7. Create a play-list. The headless sequencer can only select songs to play via a pre-configured play-list, or by placing a single song file on the command line.
8. Run `seq66cli` and then stop it (Ctrl-C) to regenerate and verify the initial configuration files, `HOME/.config/seq66/seq66cli.*`.

Once the above steps are proven for the `qseq66` configuration files, then the same settings can be made for the `seq66cli` configuration files. The easiest way to do this is to use a difference editor to make the settings match. For example:

```
$ gvimdiff ~/.config/seq66/seq66cli.rc ~/.config/seq66/qseq66.rc
```

Sometimes odd problems, such as the output synthesizer not working or not appearing in the list of outputs, can be a real puzzle. Here are the steps used in this test, based on sample files provided

in the `contrib/midi`, `data/midi`, and `data/samples` directory; adapt them to your setup. For simplicity, JACK is not running, and so ALSA is in force.

First, after booting, plug in the MIDI keyboard or MIDI control pad. Our example here will use the *Korg nanoKEY2* keyboard. (For a more powerful and exhaustive setup, see section 23 "Launchpad Mini" on page 236.)

Second, start the desired (software) synthesizer. Here we use the synth *Yoshimi*, with a stock setup from the *Yoshimi Cookbook* project. The order of starting the keyboard/pad and the synthesiser will alter the port numbers of these items. Best to do things in the same order every time... be consistent (or use the port-mapping feature described in section 21 "Port Mapping" on page 227). Run the following command in order to verify the ALSA buss numbers for the controller and the synthesizer:

```
$ aplaymidi -l       # list ALSA output ports
$ arecordmidi -l     # list ALSA input ports (except the 'announce' port)
```

Another way is to start `qseq66`, exit it, and see the results in the `qseq66.rc` file.

Third, edit the `seq66cli.rc` file as described below so that the correct settings of `[midi-clock]`, `[midi-input]` and `[midi-control-file]` are entered into the 'rc' configuration. For this discussion, we use a MIDI-control file (`nanomap.ctrl`), which we set up in the `seq66cli.rc` file to be read. The `nanomap.ctrl` file sets up the *nanoKEY2* as shown in this figure:

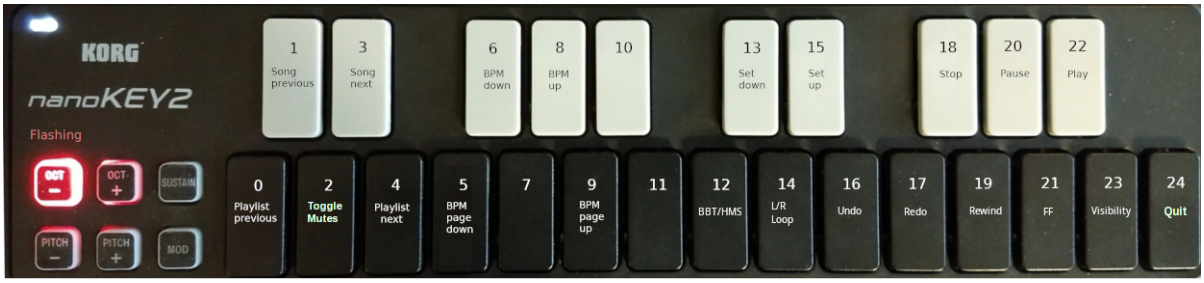


Figure 62: Sample nanoKEY2 Control Setup

In this figure, the **OCT -** button on the *nanoKEY2* is pressed until it is flashing (not seen in the figure). This means that the lowest note on the *nanoKEY2* is MIDI note 0, the lowest note possible. With these settings, the playlists and songs can be loaded, and then played and paused. The `seq66cli.rc` file is edited to specify the desired 'ctrl' file:

```
[midi-control-file]
active = true
name = "nanomap.ctrl"      # assumed to reside in ~/.config/seq66
```

The `seq66cli.rc` file should also enable the desired MIDI input control device:

```
[midi-input]
4 1    "[1] 0:1 nanoKEY2 MIDI 1"
```

This setting should match the `control-buss` setting in the the `nanomap.ctrl` file. The `nanomap.ctrl` file is included in the `data/samples` directory of the source-code package or the installation directory. The following initial settings in this file are relevant:

```
control-buss = 4 # adjust based on "aplaymidi -l"
midi-enabled = true
```

The various "midi-control-out" settings are not relevant for this test since the `nanoKEY2` cannot display statuses. For the rest of the setup, do these steps:

1. Copy the contents of of `data/seq66cli/` to `HOME/.config/seq66`.
2. Copy `data/samples/sample.playlist` and `data/samples/sample.playlist` to `HOME/.config/seq66`.
3. In your HOME directory, create a soft link to the Seq66 project (source code and data) directory: `ln -s path/to/seq66`.

Fourth, to validate the setup visibly, run a command from the command-line such as:

```
$ qseq66 --config seq66cli --buss 2 --verbose
```

The buss number ("2") may need to be different on your setup to get sound routed to the correct synthesizer. Also, the path to the playlist might need to be an absolute path; normally playlists are stored in the `HOME/.config/seq66` directory and accessed from there. Verify that the main window shows the playlist name, and that the arrow keys modify the play-list and song selection. If that works, verify that the MIDI keyboard or pad controller works to change the selection. Verify that the current song plays through the synthesizer that was started. Also verify that all songs have been directed to the desired port(s) for each song. If this setup works (MIDI controls have the proper effect and the tunes play through the synthesizer), proceed to the next step.

Fifth, test the command-line *Seq66* by running the following command (your setup might vary) on the command line:

```
$ seq66cli --buss 2 --verbose --playlist data/sample.playlist
```

There is a play-list option to automatically unmute the sets when a new song is selected. If set, then the first song should be ready to play. If it plays, and the play-list seems to work (as indicated by the console output and the proper playback), then set up to run `seq66cli` as a daemon:

```
$ seq66cli -o daemonize
```

Then it is good to solidify the setup by editing the 'rc' and 'usr' files.

One can create a shortcut to run the application in the background (i.e. not as a daemon.. In a

Fluxbox menu:

```
[exec] (Seq66 Headless) {/usr/bin/seq66cli --jack-midi --jack-master --bus 6}
```

For a sophisticated control setup, see section 23 "Launchpad Mini" on page 236. For a much more complete setup:

1. Run the seq66cli program and hit `Ctrl-C` to exit.
2. Verify that there are a number of `seq66cli.*` files in the `/.config/seq66` directory. There will also be a log-file to examine to see if all is good.
3. In `seq66cli.rc`:
    1. Verify that `[usr-file]` is active and specifies `seq66cli.usr`.
    2. Activate a `[midi-control-file]` that supports MIDI control and MIDI status display for your setup. Otherwise, the program will be nearly impossible to use.
    3. If you have mutes, playlist, and note-mapper (drums) files to use, specify them and activate them.
    4. Verify the ports and port-mapping are appropriate for both your setup and for the songs to be created or played.
    5. Verify that the JACK settings are correct (or set up to use ALSA).
4. In `seq66cli.usr`:
    1. Verify that the `[usr-midi-ppqn]` settings are as desired.
    2. Modify the `[user-options]` settings as desired.
    3. Set daemonize if desired. Might consider installing `seq66cli` as a service. We don't yet provide any real help for that use case.
    4. Change the log-file name if desired, or change it to "" to send output to /dev/null.

22.2 Seq66 Headless Stopping

If running *seq66cli* from a console, a simple `Ctrl-C` will stop the application.

If it is running as a daemon, use the following command from a console, or set up a menu entry to run this command:

```
$ killall seq66cli
```

(Not sure if it is worth adding a "–kill" option to *seq66cli*... killall is fairly complex.)

23 Launchpad Mini

This section discusses the configuration and usage of the *Novation Launchpad Mini* (we'll call it the "Mini") for control of patterns and for showing the status of *Seq66*. We will describe one of the 'ctrl' files provided with *Seq66*, the setup of ports and connections under ALSA and under JACK, and some related topics. A picture of the *Mini* appears at the end of this section. Supplemental information is documented in `contrib/notes/launchpad.txt` and `contrib/notes/launchpad-mini.ods`

(a spreadsheet).

23.1 Launchpad Mini Basics

Let's start with a guide to *Mini* programming. Some of this information was adopted from the PDF file `launchpad-programmers-reference.pdf`. That document notes that a *Mini* message is 3 bytes, and is of type Note Off (80h), Note On (90h), or a controller change (B0h). However, on our *Mini*, we do not receive Note Offs (in ALSA)... we receive Note Ons with velocity 0. The *Mini* has a top row of circular buttons numbered from 1 to 8. The next 8 rows start with 8 unlabeled square buttons on the left side with a circular button on the right, labelled with letters A through H.

The top row's circular buttons (labeled "1" through "8") emit `0xB0 cc 0x7f` on press, and `0xB0 cc 0x00` on release, where:

- **0xB0** is a Control Change on channel 0.
- **cc** is a Control Change number, ranging from 0x68 (104 decimal) to 0x6f (111 decimal) which are in the range of *undefined* MIDI controllers.

The square buttons in the 8 x 8 matrix emit `0x90 nn 0x7f` on press, and `0x90 nn 0x0` on release, where:

- **0x90** is a Note On message on channel 0.
- **nn** is the hex value of the note, as shown by the two-digit hex values shown below. The first "n" is the row number (from "0" to "7"). The second "n" is the column (from "0" to "7", and "8" for the circular buttons.

The right columns's circular buttons (labeled "A" through "H"), emit the same kind of message, with note numbers of the form `n8`.

There are two layouts available, **X-Y** and **Drum**. In *Seq66*, the drum layout is not used. See the file `contrib/notes/launchpad.txt`. X-Y Key Layout (mapping mode 1) in hexadecimal format:

```
       1     2     3     4     5     6     7     8
B0h: (68h) (69h) (6ah) (6bh) (6ch) (6dh) (6eh) (6fh)
90h: [00h] [01h] [02h] [03h] [04h] [05h] [06h] [07h] (08h) A
     [10h] [11h] [12h] [13h] [14h] [15h] [16h] [17h] (18h) B
     [20h] [21h] [22h] [23h] [24h] [25h] [26h] [27h] (28h) C
     [30h] [31h] [32h] [33h] [34h] [35h] [36h] [37h] (38h) D
     [40h] [41h] [42h] [43h] [44h] [45h] [46h] [47h] (48h) E
     [50h] [51h] [52h] [53h] [54h] [55h] [56h] [57h] (58h) F
     [60h] [61h] [62h] [63h] [64h] [65h] [66h] [67h] (68h) G
     [70h] [71h] [72h] [73h] [74h] [75h] [76h] [77h] (78h) H
```

X-Y Key Layout in decimal format:

```
       1     2     3     4     5     6     7     8
176: (104) (105) (106) (107) (108) (109) (110) (111)
```

```
144: [  0] [  1] [  2] [  3] [  4] [  5] [  6] [  7] (  8) A
     [ 16] [ 17] [ 18] [ 19] [ 20] [ 21] [ 22] [ 23] ( 24) B
     [ 32] [ 33] [ 34] [ 35] [ 36] [ 37] [ 38] [ 39] ( 40) C
     [ 48] [ 49] [ 50] [ 51] [ 52] [ 53] [ 54] [ 55] ( 56) D
     [ 64] [ 65] [ 66] [ 67] [ 68] [ 69] [ 70] [ 71] ( 72) E
     [ 80] [ 81] [ 82] [ 83] [ 84] [ 85] [ 86] [ 87] ( 88) F
     [ 96] [ 97] [ 98] [ 99] [100] [101] [102] [103] (104) G
     [112] [113] [114] [115] [116] [117] [118] [119] (120) H
```

The colors of the grid-buttons LED can be set via the command `90h key vel`, where:

- **0x90** is a Note On message on channel 0.
- **key** is a hex value given in the active of the two layouts shown above.
- **vel** is a bit mask of the form `00GGCKRR` where the bits have these meanings:
  - `GG` for Green brightness.
  - `C` to clear the LED setting of the other buffer. There are two buffers; see below for an explanation.
  - `K` to copy the data to both buffers.
  - `RR` for Red brightness.

The *Mini* has two buffers 0 and 1 which contain two separate LED states. For example, in one buffer, all LEDs can be red, and in the other buffer, all LEDs can be green. By default, buffer 0 is used for displaying and for writing. By alternating the buffers, the display can blink.

The brightness values used for green and red range from 0 (off) to 3 (full brightness). *Seq66* uses these values to provide red, green, yellow, and amber lighting.

```
Hex MSB  LSB
    00GG CKRR Color   Brightness Decimal Vel
0Ch 0000 1100 Off     Off            12
0Dh 0000 1101 Red     Low            13
0Eh 0000 1110 Red     Medium         14
0Fh 0000 1111 Red     Full           15
1Ch 0001 1100 Green   Low            28
1Dh 0001 1101 Amber   Low            29
2Ch 0010 1100 Green   Medium         44
2Eh 0010 1110 Amber   Medium         46
3Ch 0011 1100 Green   Full           60
3Eh 0011 1110 Yellow  Full           62
3Fh 0011 1111 Amber   Full           63
```

There are some other commands, not used, documented in `contrib/notes/launchpad.txt`. Also shown is a decimal version of the X-Y key layout.

We use the square grid for toggling and showing pattern muting, and also for toggling mute groups. The top row of buttons are used for *Seq66*. We start with the basic controls, mapped to the top row of circular buttons (tentative):

```
      Panic   Stop    Pause   Play    (free)    (free)  Set Dn   Set Up
Dec    68h     69h     6ah     6bh     6ch       6dh     6eh      6fh
```

```
Hex   104     105     106     107     108       109     110      111
```

The *Mini* also supports power levels, but that feature is not used by *Seq66*.

23.2 System Survey, ALSA

Let's start with ALSA. The following devices were discovered by running the commands `aconnect -lio` and `aplaymidi -l` and combining the information with the information shown on the **MIDI Clock** and **MIDI Input** tabs.

```
In  Out Port  Client name        Port name
         0:0  System             Timer
[0]      0:1  System             Announce
[1] [0] 14:0  Midi Through       Midi Through Port-0
[2] [1] 28:0  Launchpad Mini     Launchpad Mini MIDI 1       (card 3)
[3] [2] 32:0  E-MU XMidi1X1 Tab  E-MU XMidi1X1 Tab MIDI 1    (card 4)
[4] [3] 36:0  nanoKEY2           nanoKEY2 MIDI 1             (card 5)
[5] [4] 40:0  USB Midi           USB Midi MIDI 1             (card 6)
```

Note the "Timer" device, which *Seq66* does not show, and the "Announce" device, which it does show (as disabled). The device/port of interest is the `Launchpad Mini MIDI 1`, port 2 for input from the *Mini*, and port 1 for output to the *Mini*.

23.3 Control Setup

A couple of *Launchpad* control files are provided in the `/usr/share/seq66-0.93/data/linux` directory. Copy the `qseq66-lp-mini.ctrl` file to `$HOME/.config/seq66`. Make sure to exit *Seq66* before the next steps.

Open the `qseq66.rc` file. Change

```
[midi-control-file]
active = false
name = "qseq66.ctrl"
```

to

```
[midi-control-file]
active = true
name = "qseq66-lp-mini.ctrl"
```

In `qseq66-lp-mini.ctrl` or *qseq66-lp-mini-alt.ctrl*, first read through the file to get familiar with the format and purpose of this file.

23.3.1 Input Control Setup

We first want to use the *Mini* as a MIDI controller for the selection of loops, mute-groups, and various automation (user-interface) functions. In `qseq66-lp-mini.ctrl`, the only change to make for input-control is to change `0xff` to the proper *input* port. On our system, as noted above, that would be input port `[2]`.

```
[midi-control-settings]
drop-empty-controls = false           # leave as false
control-buss = 2                      # changed 0xff to 2
midi-enabled = true                   # important!
button-offset = 0                     # leave as is
button-rows = 4                       # ditto
button-columns = 8                    # ditto
keyboard-layout = qwerty              # qwertz and azerty also supported
```

Remember that `[midi-control-settings]` refers to controls *sent* to *Seq66* to control that application. Therefore, the control-buss is an **input** buss. Also remember that, in *ALSA*, *Seq66* detects and adds an "announce" buss as buss 0. This extra buss is not seen via `arecordmidi -l`, but must be accounted for... it adds 1 to the number of each input buss. It does not apply to *JACK*.

There are three sets of controls: loops, mute-groups, and automation, as described in the following sections.

23.3.1.1 [loop-control]

In the `[loop-control]` section of `qseq66-lp-mini.ctrl`, keystrokes are assigned, and only the "Toggle" (first) stanza of each MIDI control line is enabled, although there are definitions for the On and Off stanzas should one want to enable them. Here are the 32 lines, truncated. Note that they no longer include the "enabled" and "channel" columns. Instead, the event/status is checked to be non-zero in order to be enabled, and the channel is encoded in the event/status.

```
[loop-control]
 0 "1"  [ 0 0x90  0 1 127 ] [ 0 0x00  0 1 127 ] [ 0 0x00  0 1 127 ] Loop 0
 1 "q"  [ 0 0x90 16 1 127 ] [ 0 0x00  0 1 127 ] [ 0 0x00 16 1 127 ] Loop 1
 2 "a"  [ 0 0x90 32 1 127 ] [ 0 0x00 32 1 127 ] [ 0 0x00 32 1 127 ] Loop 2
 3 "z"  [ 0 0x90 48 1 127 ] [ 0 0x00 48 1 127 ] [ 0 0x00 48 1 127 ] Loop 3
 . . .
31 ","  [ 0 0x90 55 1 127 ] [ 0 0x00 55 1 127 ] [ 0 0x00 55 1 127 ] Loop 31
```

The note values (0, 16, 32, 48) are in decimal. Why? Less to type, easier to understand. The whole `[loop-control]` section is 32 lines. Also note that, above, only the "Toggle" stanza is active. The "On" and "Off" stanzas use "0x00", which disables them, even if some data values are specified.

If we want the loop armed only while the button is held, we would define something like:

```
[loop-control]
 0 "1"  [ 0 0x00  0 1 127 ] [ 0 0x90  0 1 127 ] [ 0 0x80  0 1 127 ] Loop 0
 1 "q"  [ 0 0x00 16 1 127 ] [ 0 0x90  0 1 127 ] [ 0 0x80 16 1 127 ] Loop 1
```

```
 2 "a"  [ 0 0x00 32 1 127 ] [ 0 0x90 32 1 127 ] [ 0 0x80 32 1 127 ] Loop 2
 3 "z"  [ 0 0x00 48 1 127 ] [ 0 0x90 48 1 127 ] [ 0 0x80 48 1 127 ] Loop 3
 . . .
```

The default pattern-number mapping for each *Mini* slot:

```
   1      2      3      4      5      6      7      8
[  0 ] [  4 ] [  8 ] [ 12 ] [ 16 ] [ 20 ] [ 24 ] [ 28 ] A
[  1 ] [  5 ] [  9 ] [ 13 ] [ 17 ] [ 21 ] [ 25 ] [ 29 ] B
[  2 ] [  6 ] [ 10 ] [ 14 ] [ 18 ] [ 22 ] [ 26 ] [ 30 ] C
[  3 ] [  7 ] [ 11 ] [ 15 ] [ 19 ] [ 23 ] [ 27 ] [ 31 ] D
```

By pressing the appropriate button on the *Mini*, a pattern toggles between being armed (green) and being muted (red).

Also note that there is a loop-mode setting (see section 5.1.3.5 "Pattern Loop-Record Modes" on page 70) that changes the action of each button from arm/mute to various other actions.

23.3.1.2 [mute-group-control]

The mute-group controls are very similar to the loop controls. and Off stanzas at this time; they are all zeroes.

```
[mute-group-control]
 0 "!"  [ 0 0x90  64 1 127 ] ...
 1 "Q"  [ 0 0x90  80 1 127 ] ...
 2 "A"  [ 0 0x90  96 1 127 ] ...
 3 "Z"  [ 0 0x90 112 1 127 ] ...
 . . .
31 "<"  [ 0 0x90 112 1 127 ] ...
```

The mapping is the similar to loop-control, but offset by four rows. By pressing the appropriate button on the *Mini*, a mute-group toggles between being on (selected patterns armed) and off (all patterns muted).

23.3.1.3 [automation-control]

A large number of actions available from the user-interface can also be controlled by keystrokes or a MIDI device. Here is a brief sample. See the 'ctrl' file itself for more information.

```
[automation-control]
0 "'" [ 0 0x00 0 0 0 ] [ 0 0xb0 104 127 127 ] [ 0 0xb0 104 127 127 ] # BPM Up
1 ";" [ 0 0x00 0 0 0 ] [ 0 0xb0 105 127 127 ] [ 0 0xb0 105 127 127 ] # BPM Dn
2 "]" [ 0 0x00 0 0 0 ] [ 0 0xb0   0   0   0 ] [ 0 0xb0   0   0   0 ] # Set Up
3 "[" [ 0 0x00 0 0 0 ] [ 0 0xb0   0   0   0 ] [ 0 0xb0   0   0   0 ] # Set Dn
 . . .
```

In `qseq66-lp-mini.ctrl`, all 64 square buttons are defined, which leaves the 16 circular buttons available for MIDI control. Only a few of those are defined so far.

23.3.2 Output Control Setup

Here, we want *Seq66* to send information to the *Mini* so that the lights on the *Mini* match the unmuted loops and some of the *Seq66* controls. Here are the changes to make to the output settings (while *Seq64* is *not* running). Make setting like the following:

```
[midi-control-out-settings]
set-size = 32
output-buss = 2
midi-enabled = true
button-offset = 0
button-rows = 4
button-columns = 8
```

If the device starts lighting up mysteriously while playback is happen, make sure the music is not being *played* to the control device channel. Or, if your synth makes weird noises at startup/exit, make sure the output-buss setting is not pointing to your synth. (Been there, done that! ☺)

23.3.2.1 [midi-control-out]

The `[midi-control-out]` section provides a way to see the status of each pattern/loop in the *Mini*'s grid. Here are a few entries. As per the section above, 60 is green, 15 red, 62 is yellow, and 12 is off.

```
[midi-control-out]
 0 [ 0x90   0 60 ] [ 0x90   0 15] [ 0x90   0 62] [ 0x90   0 12]
 1 [ 0x90  16 60 ] [ 0x90  16 15] [ 0x90  16 62] [ 0x90  16 12]
 2 [ 0x90  32 60 ] [ 0x90  32 15] [ 0x90  32 62] [ 0x90  32 12]
 3 [ 0x90  48 60 ] [ 0x90  48 15] [ 0x90  48 62] [ 0x90  48 12]
 . . .
```

For the `qseq66-lp-mini.ctrl` file, only the upper 32 buttons and LEDS are used for this purpose, so there are 32 lines of data in this section. The four stanzas (numbers in square brackets) are:

- **Armed**. This stanza is configured to show unmuted pattern slots as green.
- **Muted**. This stanza is configured to show muted pattern slots as red.
- **(Un)Queued**. This stanza is configured to show a queued pattern slots as yellow.
- **Empty/Deleted**. This stanza is configured to show an empty pattern slots as off (dark).

23.3.2.2 [mute-control-out]

With the `qseq66-lp-mini.ctrl` file, the lower 32 buttons can be used to see which mute-group is selected (as well as to select a mute-group). The layout is pretty simple; here are the first four of the 32 lines:

```
[mute-control-out]
1 [ 0x90  64 60 ] [ 0x90  64 15 ] [ 0x90  64 12 ]
2 [ 0x90  80 60 ] [ 0x90  80 15 ] [ 0x90  80 12 ]
3 [ 0x90  96 60 ] [ 0x90  96 15 ] [ 0x90  96 12 ]
4 [ 0x90 112 60 ] [ 0x90 112 15 ] [ 0x90 112 12 ]
. . .
31 [ 0x00   0  0 ] [ 0x00   0  0 ] [ 0x00   0  0 ]
```

The slots are numbered; all of the entries in the section are always enabled. The first stanza indicates that the button the selected mute-group will be green. The second stanza indicates that the unselected mute-group buttons will all be red, as long as they have mutes defined in them. The third stanza indicates that the inactive (empty) mute-groups will be dark.

23.3.2.3 [automation-control-out]

This section allows for the following status to be shown. "0xb0" indicates a circular button in the top row. A "1" at the beginning of each line indicates that output is active.

```
[automation-control-out]
1 [ 0xb0 104  60 ] [ 0xb0 104  15 ] [ 0xb0 104  12 ]  # Panic
0 [ 0x00   0   0 ] [ 0x00   0   0 ] [ 0x00   0   0 ]  # Slot_shift
1 [ 0x90   8  60 ] [ 0x90   8  15 ] [ 0x90   8  12 ]  # Queue
. . . and many more
```

Note that the play, stop, and pause statuses are all shown on the same button, as green, red, or yellow. Some of these might still be in progress as you read this.

The stanzas mostly show the coloring for on (60 = green), off (15 = red), and inactive/unconfigured (12 = off). Some of the buttons provide other meanings. Here is the complete table. The color is indicated by R, G, Y, or A, while "Inactive" is the most common entry for the third stanza and represents a darkened button. An number indicates we don't like something about the behavior and might fix it, or explain it. See the notes about those after the table. For example, currently, pressing "Panic" causes the **Panic** and **Stop buttons** to turn green, which seems silly in hindsight.

Table 15: Status Announcement Stanzas

Action	**Stanza 1**	**Stanza 2**	**Stanza 3**
Panic 1	On (R)	Off (R)	Inactive
Stop 1	On (R)	Off (R)	Inactive
Pause 1	On (Y)	Off (R)	Inactive
Play 1	On (G)	Off (R)	Inactive
Toggle Mutes 2	On (G)	Off (R)	Inactive
Song Record 3	On (G)	Off (R)	Inactive
Queue 4	On (G)	Off (R)	Inactive
One-short 4	On (G)	Off (R)	Inactive
Replace 4	On (G)	Off (R)	Inactive
Snapshot 4	On (G)	Off (R)	Inactive
Song-mode 3	On (G)	Off (R)	Inactive
Learn 3	On (G)	Off (R)	Inactive
BPM-Up 5	On (G)	Off (G)	Inactive
BPM-Dn 5	On (G)	Off (Y)	Inactive
List-Up 5	On (G)	Off (G)	Inactive
List-Dn 5	On (G)	Off (Y)	Inactive
Song-Up 5	On (G)	Off (G)	Inactive
Song-Dn 5	On (G)	Off (Y)	Inactive
Set-Up 5	On (G)	Off (G)	Inactive
Set-Dn 5	On (G)	Off (Y)	Inactive
Tap_BPM	On (G)	Off (G)	Inactive
Quit	On (R)	Off (R)	Inactive
Visibility	On (A)	Off (A)	Inactive

Notes (*):

1. *Panic*, *Stop*, *Pause*, and *Play*. These needed improved coordination and appearance. So Panic and Stop are now always in "off" status.
2. *Toggle Mutes*. There's no easy way to detect this transition, but this status is reflected in the loop buttons and is obvious. We could consider just toggling the color between two colors.
3. *Song Record*, *Slot Shift*, *Song-mode*, *Learn*. These are state indicators, so it is green when in that state, and red when not in that state.
4. *Queue*, *One-shot*, *One-shot*, *Replace*. These operate by striking the appropriate key and then selecting the pattern hot-key to effect. So we could turn the key green until the hot-key/pattern is selected.
5. *BPM-Up*, *BPM-Dn*, *List-Up*, *List-Dn*, *Song-Up*, *Song-Dn*, *Set-Up*, *Set-Dn*. These are up-down pairs. We currently just show them in red, but probably more informative to show Up in green and Dn in yellow.

23.4 Test Run, ALSA

Now that we're set up, start *Seq66*.

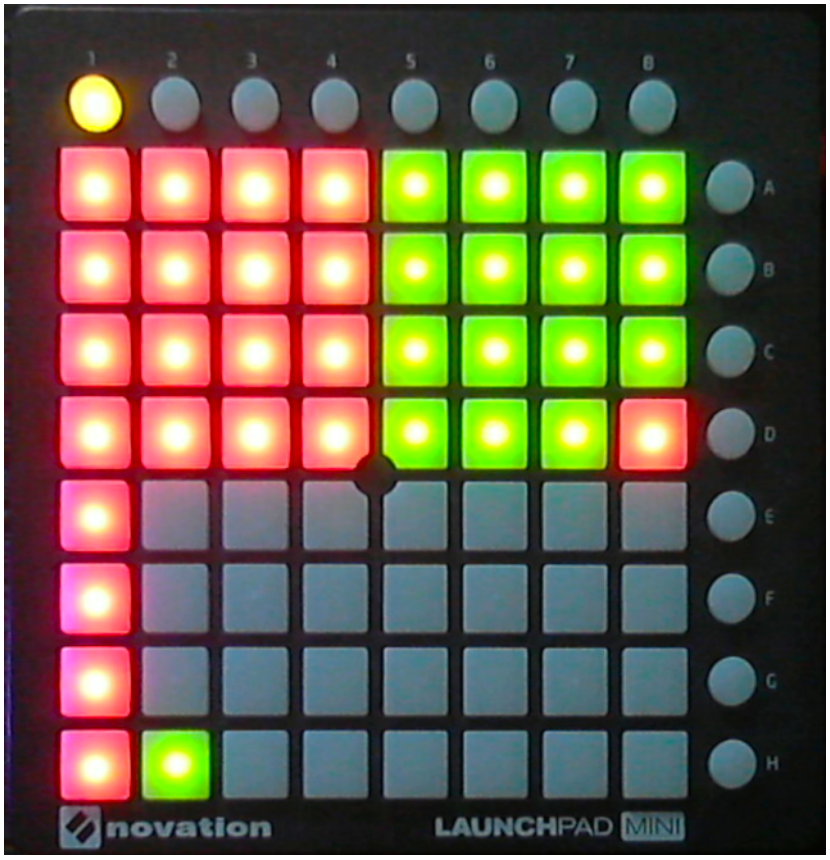


Figure 63: Launchpad Mini Running with Seq66

This picture shows that playback is paused (yellow), that mute-group 7 is active, and that all the patterns in that mute-group are green, except for one that got muted accidentally while taking the picture. If the **File** / **New** option is selected, all the patterns are turned off, but the four mute-group buttons at the bottom left remain, as the mute-groups are not erased. (Bug or feature?) What's next? First, add more controls and statuses to the configuration. Second, start working on a MIDI file to produce a light show!

23.5 System Survey, JACK

Now let's see what we have to do in *JACK*. First peruse section 20 "JACK" on page 220, to understand the basics about *JACK*, including the last section there that describes how to set up *ALSA*-to-*JACK* bridging. Run one of the following commands (they are identical in function), then verify the ports in **Edit** / **Preferences** / **MIDI Clock** and **MIDI Input**, and then exit.

```
$ qseq66 --jack-midi
$ qseq66 --jack
```

In `qseq66.rc`, in section `[jack-transport]`, one will find this setting:

```
jack-midi = true
```

In `qseq66.rc`, in section `[midi-input]`, the MIDI inputs are shown, decorated with the "a2j" designation:

```
[midi-input]
5   # number of input MIDI busses
0 1 "[0] 0:0 seq66:a2j Midi Through [14] (capture): Midi Through Port-0"
1 0 "[1] 0:1 a2j:Launchpad Mini [28] (capture): Launchpad Mini MIDI 1"
2 0 "[2] 0:2 a2j:E-MU XMidi1X1 Tab [32] (capture): E-MU XMidi1X1 ..."
3 0 "[3] 0:3 a2j:nanoKEY2 [36] (capture): nanoKEY2 MIDI 1"
4 0 "[4] 0:4 a2j:USB Midi [40] (capture): USB Midi MIDI 1"
```

This is very similar to the *ALSA* setup, except that there is *no* "announce" port in *JACK*. The *Mini*'s input buss has shifted from port 2 to port 1. And, of course, the port names are a lot longer. Similarly, for the MIDI outputs:

```
[midi-clock]
5    # number of MIDI clocks (output busses)
0 0  "[0] 0:0 seq66:a2j Midi Through [14] (playback): Midi Through. . ."
1 0  "[1] 0:1 seq66:a2j Launchpad Mini [28] (playback): Launchpad. . ."
2 0  "[2] 0:2 seq66:a2j E-MU XMidi1X1 Tab [32] (playback): E-MU . . ."
3 0  "[3] 0:3 seq66:a2j nanoKEY2 [36] (playback): nanoKEY2 MIDI 1"
4 0  "[4] 0:4 seq66:a2j USB Midi [40] (playback): USB Midi MIDI 1"
```

We make sure that the correct `control-buss` and `output-buss` are set, and both have the setting `midi-enabled = true` in `qseq66-lp-mini.ctrl`. Then make sure that `qseq66.rc` has its `[midi-control-file]` set to:

```
"qseq66-lp-mini.ctrl"
```

Run `qseq66` again, and make sure that the *Mini*'s input and output ports are enabled. (Unfortunately, if one has to enable them, the application will need to be restarted.) The results should be just like section 23.4 "Test Run, ALSA" on page 244.

24 Concepts

The *Seq66* program is a loop-player machine with a number of interfaces. This section is useful to present some concepts and definitions of terms as they are used in *Seq66*. Various terms have been used over the years to mean the same thing (e.g. "sequence", "pattern", "loop", "track", and "slot"), so it is good to clarify the terminology.

24.1 Concepts / Reload Session

While *Seq66* is flexible, changes in devices setups and other settings generally required the user to *reload session*. This reload can be done in two ways.

- **Manual Restart** Once a setting is changed, or a new arrangement of devices occurs, exit *Seq66*, examine or edit the various configuration files (if desired), and start *Seq66* again.
- **Edit / Preferences / Restart Seq66**. When settings in **Edit / Preferences** are changed, the **Reload Session** button becomes enabled. Click it, and the result is basically like restarting *Seq66*.

Note that some changes in settings will not take effect on this restart. For example, a style-sheet, if specified, is loaded when the application is created, not in the restartable loop. The solution is to exit *Seq66* manually and then start it again.

In the future (version 2?), we will make *Seq66* able to better detect and reconfigure for system changes and preferences changes.

24.2 Concepts / Terms

This section doesn't provide comprehensive coverage of terms. It covers terms that might be puzzling.

24.2.1 Concepts / Terms / loop, pattern, track, sequence

Loop is a synonym for *pattern*, *track*, or *sequence*; the terms are used interchangeably. Each loop is represented by a box (pattern slot) in the Pattern (main) window, also known as the "Patterns Panel" or "Live Grid"..

A loop is a unit of melody or rhythm extending for a small number of measures (in most cases). Each loop is represented by a box in the patterns panel. Each loop is editable. All patterns can be layed out in a particular arrangement to generate a more complex song. A *slot* is a box in a pattern grid that holds a loop. Note that other sequencer applications use the term "sequence" to apply to the complete song, and not just to one track or pattern in the entire song.

24.2.2 Concepts / Terms / armed, muted

An armed sequence is a MIDI pattern that will be heard. "Armed" is the opposite of "muted", and the same as "unmuted". An unarmed sequence will not be heard, and it has a normal background. Performing an *arm* operation in *Seq66* means clicking on an "unarmed" sequence in the patterns panel (the main window of *Seq66*). When the sequence is *armed*, it will be heard, and it has a more noticeable background. A sequence can be armed or unarmed in many ways:

- Clicking on a sequence/pattern box.
- Pressing the hot-key for that sequence/pattern box.

- Setting up a mute-group containing that pattern, and then activating it with its hot-key.
- Opening up the Song Editor and starting playback; the sequences arm/unarm depending on the layout of the sequences and triggers in the piano roll of the Song Editor.
- Using a MIDI control, as configured in a 'ctrl' file, to toggle the armed status of a pattern or select a mute-group.

24.2.3 Concepts / Terms / bank, screenset, play-screen

A *screenset* is a set of patterns that fit within the **4 x 8** grid of loops/patterns in the patterns panel. *Seq66* supports multiple screens sets, up to 32 of them, and a name can be given to each for clarity. Some other sizes, such as **8 x 8** and **12 x 8**, are partly supported. The term "bank" is *Kepler34*'s name for "screenset". The term "scenes" from *Ableton Live* is very similar as well. By default, only one set is active and playing at a time. This set is informally termed the "play screen". There are options to let more than one set play at once.

24.2.4 Concepts / Terms / buss, bus, port

A *buss* (also spelled "bus" these days, see https://en.wikipedia.org/wiki/Busbar) is an entity onto which MIDI events can be placed, in order to be heard or to affect the playback, or into which MIDI events can be received, for recording. A *buss* is just another name for port. *Seq66* can also perform some mapping of I/O ports for a more flexible studio setup.

24.2.5 Concepts / Terms / performance, song, trigger

In the jargon of *Seq66*, a *performance* or *song* is an organized collection of patterns that play a full tune automatically. This layout of patterns is created using the song editor, sometimes called the "performance editor". This window controls the song playback in "Song Mode" (as opposed to "Live Mode").

The playback of each track is controlled by a set of triggers created for that track. A *trigger* is indicates when a sequence/pattern/loop should be played, and how much of the sequence (including repeats) should be played. A song performance consists of a number of sequences, each triggered as the musician laid them out.

24.2.6 Concepts / Terms / Auto-step, Step-Edit

Auto-step (step-edit) provides a way to add notes easily when a pattern is not playing. It works in two ways. In the first way, when drawing notes on the pattern-editor's piano roll, dragging the mouse automatically inserts notes of the configured length at intervals of the configured snap. In the second way, incoming MIDI notes (including chords) are automatically logged at the given snap interval, with a length a little less than the configured snap interval. There is also a one-shot step-edit mode that is useful for recording a drum pattern emitted by a machine.

24.2.7 Concepts / Terms / export

A *export* in *Seq66* is a way of writing a song-performance to a more standard MIDI file, so that it can be played exactly by other sequencers. An export collects all of the unmuted tracks that have performance information (triggers) associated with them, and creates one larger trigger for each track, repeating the events as indicated by the original performance. MIDI can also be exported to SMF 0 format.

24.2.8 Concepts / Terms / group, mute-group

A *group* in *Seq66* is a set of patterns, that can arm/disarm all together. Every group contains all sequences in the active screen set. This concept is similar to mute/unmute groups in hardware sequencers. Also known as a "mute-group". Mute-groups can be stored in the MIDI file or in a 'mutes' file. Each mute-group is associated with a keystroke or a MIDI control.

24.2.9 Concepts / Terms / play-set

A *play-set* is the current set of patterns that are available for playing. Normally, this is the patterns in the current bank (play-screen), but *Seq66* can be configured (see section 12.3 "'rc' File" on page 141) to include patterns from other sets. For example, one can have each newly selected set get added to the play-set without removing the previous set's patterns from the play-set.

24.2.10 Concepts / Terms / PPQN, pulses, ticks, clocks, divisions

The concept of "pulses per quarter note", or PPQN, is very important for MIDI timing. To make it a bit more confusing, sometimes these pulses are referred to as "ticks", "clocks", and "divisions". To make it even more confusing, there are separate timing concepts to understand, such as "tempo", "beats per measure", "beats per minute", and "MIDI clocks". And, when JACK is involved, one must remember that JACK "ticks" come at 10 times the rate of MIDI ticks. A full description of all these terms, and how they are calculated, is beyond the scope of this document.

24.2.11 Concepts / Terms / queue, keep queue, snapshot, oneshot

The concepts here refer to playback, not recording. The term "one-shot" in playback is differnt from the same term in "recording".

To "queue" a pattern means to ready it for playback on the next repeat of a pattern. A pattern can be armed immediately, or it can be queued to play back the next time the pattern restarts. Pattern toggle occurs at the next play of the pattern, rather than being set immediately.

A set of queued patterns can be temporarily stored, so that a different set of playbacks can occur, before the original set of playbacks is restored.

The "keep queue" functionality allows the queue to be held without holding down a button the whole time. Once this key is pressed, then the hot-keys for any pattern can be pressed, over and

over, to queue each pattern.

A *Seq66 snapshot* is a briefly preserved state of the patterns. One can press a snapshot key, change the state of the patterns for live playback, and then release the snapshot key to revert to the state when it was first pressed. (One might call it a "revert" key.)

Seq66 one-shot playback is a way to enable a pattern to be queued and then played only once. For recording, *Seq66 one-shot recording* is a way to record notes from a MIDI device with automatic stepping and snapping, which terminates at the end of the specified length of the pattern.

24.3 Concepts / Sound Subsystems

24.3.1 Concepts / Sound Subsystems / ALSA

ALSA is a audio/MIDI system for *Linux*, with components built into the *Linux* kernel. It is the main subsystem used by *Seq66*. It supports virtual port connections via the `aconnect` program. The name of the library used to build *ALSA* projects is `libasound` [1].

24.3.2 Concepts / Sound Subsystems / PortMIDI

PortMIDI is a cross-platform API (applications programming interface) for MIDI refactored for *Seq66*. It is used in the "portmidi" C++ modules, and provides support for *Seq66* in *Microsoft Windows* (and potentially *Mac OSX*). See reference [27] for the PortMIDI home page; our version cuts out code that requires *Java*.

24.3.3 Concepts / Sound Subsystems / JACK

JACK is a cross-platform API and infrastructure (with an emphasis on *Linux*) to make it easier to connect and reroute MIDI and audio event between various applications and hardware ports. It should be preferred over *ALSA*, and is selected automatically if running. It supports virtual port connections via the `qjackctl` program or the *Non Session Manager*. See reference *JACK Audio Connection Kit* [7].

25 MIDI Format and Other MIDI Notes

Seq66 tries to handle most MIDI files, and to provide song information in a format that does not break other MIDI-compliant sequencers. It can read SMF 0, 1, and *Cakewalk* WRK files. Another format, SMF 2, is a series of SMF 0 files for multiple songs. The MIDI data is stored in separate tracks, which are additionally wrapped in containers, so it's possible to have several tracks using the same MIDI channels. SMF 2 never caught on, and *Seq66* does not support it.

MIDI SMF 0 files have all information on one track, which mixes together data on all channels of events included in the songe. MIDI SMF 1 files have channel data on separate tracks. *Seq66* can use this channel information, but its main mode is to ignore the channel information and replace it

with the channel requested by the user. *Seq66* also embeds extra information into a song via the "Sequencer-Specific Meta-Event" mechanism described on page 139 of the following document:

http://www.shclemen.com/download/The Complete MIDI1.0 Detailed Spec.pdf

25.1 Standard MIDI Format

25.1.1 Standard MIDI Format 0

Seq66 can read and import SMF 0 MIDI files, and splits the MIDI data into separate tracks by channel number. When SMF 0 format is detected, *Seq66* puts all of the events into the same sequence/pattern, pattern 16. As the file is processed, a list of the channels present in the track is maintained.

Once the end-of-track is encountered in the SMF 0 file, a new empty slot is created for each channel found. The events in the main pattern are scanned and added to the the appropriate pattern. If the event is a channel event, then the event is inserted into the pattern that was created for that channel. If the event is a non-channel event, then each pattern gets a copy of that event.

After processing, the MIDI buss information, track name, and other pieces of information are attached to each sequence. The imported SMF 0 track is preserved, in pattern slot #16. One can delete this track before saving the file, or just keep it muted.

The sequence number of each new track is the internal channel number (always the MIDI channel number, minus one). The time-signature of each track is set to the default, unless a time-signature event is encountered in the imported file.

Tempo and Time Signature events are read, if present. When saving a *Seq66* MIDI file, the Tempo and Time Signature events are saved as MIDI events in the first track. This allows other sequencers to read a *Seq66* MIDI file.

An example of an SMF 0 file, `CountryStrum.mid`, is included with the source code in the `contrib/midi` directory. `CountryStrum.midi` is the SMF 1 version of this file converted by *Seq66*.

25.2 Sequencer-Specific Meta-Events Format

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.

The authors of *Seq24* took trouble to ensure that the format of the MIDI files it writes are compatible with other MIDI applications. *Seq24* also stores its "proprietary" (an unfortunate legacy term) information (triggers, mute-groups, MIDI control information, etc.) in the file, but marked as per the MIDI specification so that other sequencers can read the file and ignore its *Seq24*-specific information. *Seq66* continues that MIDI-compliant behavior, and improves it. Each sequence/pattern/loop can contain special information, such as the palette color assigned to that track.

In *Seq24*, some these events are placed at the end of the song, and some are included in each pattern. Most MIDI applications handle this situation fine, but some (e.g. midicvt) do not. *Seq66* 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 *Seq66*) 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.

```
FF 7F len id data
```

The first byte of the message is the "manufacturer ID", which *Seq24* set to "24". (This value corresponds to the manufactuer ID of *Hohner*!) Another "24" is added to make the number `0x2424` easy to search in a binary hex editor, such as *hexer* or *bvi*. Then a "00" is added. Finally, the last number, "nn" is added, and that specifies the type of data to be read. Here is the full encoded format:

```
delta FF 7F len 24 24 00 nn databyte(s)
```

The "len" value includes the 4 bytes of the 0x242400nn SeqSpec marker and the number of data bytes. If the tag value "nn" is not recognized, a message is emitted and the `SeqSpec` is skipped. If the "24" value is something else, as it would be for another sequencer product, then the `SeqSpec` is skipped silently.

All of the `SeqSpecs` are shown in the next table. It shows the name, length, and data for each one. A length of 0 means the `SeqSpec` is not implemented. The `c_triggers` tag is obsolete, but still present, for backward compatibility. A named value (e.g. "buss") indicates a byte that specifies the value set by the `SeqSpec` . Please note that some of these values (for example, `c_timesig` could be better implemented by standard MIDI meta-events. Legacy code! Some `SeqSpec` sections appear only if they are set to a non-default value. For example, patterns having no color would not likely have a `c_seq_color SeqSpec`. Flags are always one byte; the minimum "length" value is 5. Values with more than one byte are indicated by "S" (short, or two bytes), or "L" (long, or four bytes).

An asterisk indicates a per-track setting, as opposed to a whole-song setting. A question-mark indicates that the tag is either deprecated or not yet implemented. For example, `c_midictrl` is completely replaced by the 'ctrl' file, though *Seq66* will still read (and ignore) this `SeqSpec` if present. A `SeqSpec` named "gap" or "reserved" is not used.

Note that the base length is 4 bytes, the size of a `0x242400nn` value. Also note that some of the multi-byte (2 or 4 bytes) values that indicate counts are stored in big-endian (network order) format. The most-signifant byte is grabbed first, then left-shifted to get read for the next significant bit. See the functions `midifile::read_short()` and `midifile::read_long()`, and contrast them with `midifile::read_varinum()`.

25.2.1 SeqSpec c_midibus

`c_midibus`: specifies the desired output buss/port number for a track.
Length: 5
Format: `0x24240001 buss`

Table 16: All SeqSpec Items

SeqSpec tag	Type	Length	Data
c_midibus	Track	5	0x24240001 buss
c_midichannel	Track	5	0x24240002 channel
c_midiclocks	Unused	4+count	0x24240003 count:L bussclocks
c_triggers	TBD	0	0x24240004 (see c_triggers_ex)
c_notes	TBD	2+variable	0x24240005 setcount strings...
c_timesig	Track	6	0x24240006 bpb bw
c_bpmtag	TBD	8	0x24240007 bpm:L
c_triggers_ex	Track	4+triggercount*12	0x24240008 triggers...
c_mutegroups	Global	4+4*groups+4*seqs	0x24240009 groups:S seqs:S data...
c_gap_A	Unused	4	0x2424000A
c_gap_B	Unused	4	0x2424000B
c_gap_C	Unused	4	0x2424000C
c_gap_D	Unused	4	0x2424000D
c_gap_E	Unused	4	0x2424000E
c_gap_F	Unused	4	0x2424000F
c_midictrl	TBD	4+8*ctrls	0x24240010 ctrls data...
c_musickey	Both	5	0x24240011 key
c_musicscale	Both	5	0x24240012 scale
c_backsequence	Both	8	0x24240013 seqnumber:L
c_transpose	Track	5	0x24240014 transpose
c_perf_bp_mes	Global	8	0x24240015 bpb:L
c_perf_bw	Global	0	0x24240016 bw:L
c_tempo_map	Seq32	0	0x24240017
c_midiinbus	Track	5	0x24240018
c_reserved_2	TBD	0	0x24240019
c_tempo_track	Global	8	0x2424001A track:L
c_seq_color	Track	5	0x2424001B color
c_seq_edit_mode	Kepler34	0	0x2424001C
c_seq_loopcount	Track	6	0x2424001D 00 00
c_reserved_3	TBD	0	0x2424001E
c_reserved_4	TBD	0	0x2424001F
c_trig_transpose	Track	4+triggercount*(12+1)	0x24240020 triggers...

The output buss value for a track can be set in the pattern editor or using the buss-dropdown in the main window. The buss value ranges from 0 to 47, though the practical range is under a dozen ports. Currently in *Seq66*, buss information is stored by number only. However, there is a port-mapping mechanism in the 'rc' file that let's one assign permanent port numbers by name, and translate those to actual port numbers. See section 21 "Port Mapping" on page 227.

25.2.2 SeqSpec c_midiinbus

`c_midiinbus`: specifies the optional desired input buss/port number for a track.
Length: 5
Format: `0x24240018 buss`

This value is similar to the output-buss specification in the previous section. It sets an input bus for the pattern. When recording, each pattern that specifies an input bus will receive MIDI events from that buss, so that multiple devices can be recorded at once. Only the first buss having that number receives the MIDI events. If there is no input buss specified, which is the normal case, then the input is considered "Free".

25.2.3 SeqSpec c_midichannel

`c_midichannel`: specifies the desired output channel number for a track.
Length: 5
Format: `0x24240002 channel`

The channel value for a track can be set in the pattern editor. This channel value ranges from 0 to 15, and has an "null" value of 0x80. If the channel ranges from 0 to 15, that channel is applied to every channel event that goes out from that sequence. Otherwise, the channel of the MIDI event is used for output. Note that SMF 0 MIDI files have a single track and can have a mixture of different channels in its channel event. If *Seq66* detects SMF 0, it splits the channel events into different patterns. Also note that "channel" 0x80 is a flag to use the channel stored in each channel event. This value represents the "Free" setting in the pattern editor.

25.2.4 SeqSpec c_midiclocks

`c_midiclocks`: specifies the clocking for the busses, but is inactive.
Length: 8
Format: `0x24240003 count:L bussclocks`

This item is a hold-over from `Seq24`. It was meant, presumably, to hold the clocking statuses of the output busses. However, it seems to have fallen by the wayside, and the only item read/written is 4 bytes of zeroes. Clocking is specified in the 'rc' file, and can be edited in the preferences dialog. See section 12.3.6 "'rc' File / MIDI-Clock Section" on page 146.

25.2.5 SeqSpec c_triggers

`c_triggers`: specifies the old format for song-performance triggers.
Length: Indeterminate
Format: `0x24240004 buss`

This `SeqSpec` is no longer used, but is still read and converted to a valid trigger, encountered. We have not encountered a file containing this value yet. Instead, `c_triggers_ex` is used.

25.2.6 SeqSpec c_notes

`c_notes`: specifies the names of the sets in the tune.

Length: 2 + variable
Format: `0x24240005 setcount [ length string ] [ ... ]`

Each set in a tune can have a name. The first value is a 2-byte value indicating the number of sets in the tune. This value can range from 0 to 31, for a total of 32 sets. *Seq66* limits the number of sets to 32 for practical reasons. See section 14 "Seq66 Set Master" on page 192.

After the set-count comes the list of set-name segments. Each segment starts with a 2-byte value indicating the size of the string, followed by that number of bytes for the text of the string. Presumably, the text must be in either ASCII encoding or UTF-8 encoding.

25.2.7 SeqSpec c_timesig

`c_timesig`: specifies the time signature for a track.
Length: 6
Format: `0x24240006 bpb bw`

This `SeqSpec` specifies the underlyng time-signature for the track in a 2-byte format, the beats-per-bar followed by the beat-width. This time-signature is meant for setting up the pattern editor to the user's preferences, and is not the time-signature for the song. It is *not* set globally in the `seq66::performer` class; that can be done in the main window.

When a time-signature is added (see section 6.1 "Pattern Editor / First Row" on page 76, how it is stored depends if it is a MIDI-standard power-of-2 beat-width or not. If a power of 2, a time-signature event is added. Otherwise, the beat-width is set for the pattern, and is saved as the `c_timesig` event.

The normal time-signature meta-event is first obtained from values stored in the performer class, and has the format `0xFF 58 4 bpb bw cpm tpq`, where cpm is the clocks-per-metronome and tpq is the 32nds-per-quarter.

25.2.8 SeqSpec c_bpmtag

`c_bpmtag`: specifies the the BPM in a format allowing double precision.
Length: 8
Format: `0x24240007 bpm:L`

Due to requests for higher precision in the beats-per-minute of a song, this value is the value of the BPM multiplied by 1000. When read, it is divided by 1000 to get the desired floating-point precision.

The normal tempo meta-event format is `0xFF 51 03 tttttt`, where "tttttt" is 3 bytes representing the number of microseconds per quarter note. The function `tempo_us_from_bytes()` calculates the microseconds from these three bytes; the "inverse" function is `tempo_us_to_bytes()`.

Generally, the MIDI tempo comes first in the file, and the `SeqSpec` tempo comes later. The last value obtained is the BPM that the performer module contains. The conversion between the

`SeqSpec` format and the MIDI Tempo format is effected by the functions `bpm_from_tempo_us()` and `tempo_us_from_bpm()`.

25.2.9 SeqSpec c_triggers_ex

`c_triggers_ex`: specifies the triggers for a given track.
Length: 4 + triggercount * 12
Format: `0x24240008 [trigger-on off offset ] [ ... ]`

The triggers in each pattern in a song determines the layout of the song in the **Song Editor**. The extent of each trigger is partly determined by the PPQN of the song and whether or not PPQN rescaling is needed (when PPQN != 192).

The number of triggers is determined by dividing the `SeqSpec len` value by the size of a trigger, 12 bytes. The format of each trigger is three 4-byte (long) values: `on:L off:L offset:L`.

Each trigger is represented by an `seq66::trigger` object. Each trigger has a start and an end tick value based on the recorded pattern loop, and an offset value that indicates how much the trigger is delayed as laid out in the song editor.

25.2.10 SeqSpec c_trig_transpose

`c_trig_transpose`: specifies the triggers for a given track, plus a transposition value.
Length: 4 + triggercount * (12 + 1)
Format: `0x24240020 [trigger-on off offset transpose] [ ... ]`

This is an extension to `c_triggers_ex` that adds a value to use to transpose the pattern at this particular trigger. Very useful for simple repetitive patterns like that in *Kraftwerk's "Europe Endless"*. If the trigger has a zero tranpose value, then `c_trig_transpose` is still written, but the extra byte is zero.

In order to preserve some of the ability of older sequencers to read this section, if all of the triggers are non-transposed, then the old-style triggers () are written for that pattern. The older trigger tags will be read if present in the *Seq66* MIDI file.

25.2.11 SeqSpec c_mutegroups

Legacy format (for *Seq24* and early versions of *Seq66*:

`c_mutegroups`: specifies the mute-groups in play in a song. The group and sequence counts are encoding in a "split-long" format (denoted by "S") compatible with the legacy (*Seq24*) format.
Length: 4 + 4 * groupcount + 4 * seqcount
Format: `0x24240009 groupcount:S seqcount:S groupdata...`

For newer versions of *Seq66*, a space-saving format (over *Seq24*) is used, which uses 1 byte, instead of 4, to encode each mute status.

`c_mutegroups`: As above, except that it includes an optional name for each group; the total size of these names is the length of each name plus the surrounding quote characters (namecount). The group and sequence counts are encoding in a "split-long" format (denoted by "S") compatible with the legacy (*Seq24*) format.
Length: 4 + 1 * groupcount + 1 * seqcount + namecount
Format: `0x24240009 groupcount:S seqcount:S groupdata...`

In the new format, we combine the group-count and pattern-count into one long value. This value is split, when read, into these counts. By default, the groupdata is formatted as follows, where all data items are encoded as bytes:

```
<Group number>
<32 values of 0 or 1 for mute statuses>
Optional: <"name...">
```

Mute-groups enable the toggling of arming/muting for an arbitrary set of multiple patterns at once. *Seq66* supports up to 32 mute-groups of size 32 patterns. More flexibility is planned.

Mute-groups can also be stored in a 'mutes' files, which is probably a better place to store them if one uses a consistent setup for all one's tunes. (See section 12.6 "'mutes' File" on page 181, and section 53 "Mute Master Tab" on page 196). Here, we describe how they are encoded in the song.

The first two values provide the group-count and the sequence-count. Then, the groups are looped through. Each group has the format `groupnumber bits` where "bits" is a string of up to thirty-two 4-byte (!) values indicating if the corresponding pattern is part of the group. This setup is a real space waster, and in newer versions of *Seq66*, the MIDI file encodes mute-groups using byte-size values.

25.2.12 SeqSpec c_gap_(ABCDEF)

This set of six `SeqSpec` values is a gap created by skipping from `0x24240009` to `0x24240010` as if the numbers were decimal, a long-standing oversight from *Seq24*. We guarantee that *Seq66* will never use these values.

25.2.13 SeqSpec c_midictrl

`c_midictrl`: specifies the MIDI controls to be used.
Length: 4 + 8 * ctrls
Format: `0x24240010 ctrls data...`

This section apparently provided a way to save MIDI controls for loops (toggle, on, and off) inside the song. However, it makes more sense to save them in the 'ctrl' file. This `SeqSpec` is parsed

if present, but the data is thrown away, and this `SeqSpec` is never written. Oddly enough, *Seq24* would read this section, but would write only four bytes of zeroes.

25.2.14 SeqSpec c_musickey

`c_musickey`: specifies the musical key for the song.
Length: 5
Format: `0x24240011 key`

The value of `key` ranges from 0 ("C") to 11 ("B"). This item specifies the musical key for a song (globally), but it can also be specified inside each pattern, as well, so that patterns can have different keys. When provided globally, this option is stored in the `seq66::usrsettings` class.

25.2.15 SeqSpec c_musicscale

`c_musicscale`: specifies the musical scale for the song.
Length: 5
Format: `0x24240012 scale`

The value of `scale` ranges from 0 ("Off") to 8 ("Minor Pentatonic""). This item specifies the musical scale for a song (globally), but it can also be specified inside each pattern, as well, so that patterns can have different scales. When provided globally, this option is stored for the duration of the session in the `seq66::usrsettings` class.

25.2.16 SeqSpec c_backsequence

`c_backsequence`: specifies the background sequence to be shown.
Length: 8
Format: `0x24240013 backsequence:L`

This item specifies the background sequence to display in the pattern editor for a song (globally), but it can also be specified inside each pattern, as well, so that patterns can show different background sequences. When provided globally, this option is stored for the duration of the session in the `seq66::usrsettings` class.

25.2.17 SeqSpec c_transpose

`c_transpose`: specifies if a pattern can be transposed.
Length: 5
Format: `0x24240014 transposable`

This `SeqSpec` applies to patterns only. Unlike `c_trig_transpose`, it applies not to the triggers, but to the whole pattern, and is merely a boolean value. It is set to a non-zero value (1) to indicate

that a pattern can be transposed, either on the fly or via the note-mapping features (see section 12.7 "'drums' File" on page 184). A value of 0 is useful to mark a drum pattern and prevent it from being transposed.

A value of 0 also prevents the drum pattern from being note-mapped. See section 12.3.5 "'rc' File / Note Mapper" on page 145. The user must temporarily enable transposition in the pattern editor, and then press the **Map**. This should be done only once, otherwise the drum pattern will sound like a random set of percussive instruments. This section is always saved with the pattern.

25.2.18 SeqSpec c_perf_bp_mes

`c_perf_bp_mes`: specifies the beats-per-bar for the performance.
Length: 5
Format: `0x24240015 bpb:L`

Provides an override for the beats-per-bar from the Time Signature in track 0. Note that the beats-per-bar is currently settable from this value, a true MIDI time-signature event, and `c_timesig`! This issue needs to be cleaned up.

25.2.19 SeqSpec c_perf_bw

`c_perf_bw`: specifies the beat-width for the performance.
Length: 5
Format: `0x24240016 bpb:L`

Provides an override for the beat-width from the Time Signature in track 0. Note that the beats-per-bar is currently settable from this value, a true MIDI time-signature event, and `c_timesig`! This issue needs to be cleaned up.

25.2.20 SeqSpec c_tempo_map

`c_tempo_map`: Not implemented.
Length: 5
Format: `0x24240017`

This section is an unimplemented *Seq32* feature. Tempo events can be added to pattern 0.

25.2.21 SeqSpec c_reserved_(12)

`c_reserved_2`: Reserved for expansion.
Length: Indeterminate
Format: `0x24240019`

25.2.22 SeqSpec c_tempo_track

`c_tempo_track`: specifies the alternate tempo track for a song.
Length: 8
Format: `0x2424001A track:L`

Normally, the song tempo should be stored in the first track. This value can be set to move it to another pattern.

25.2.23 SeqSpec c_seq_color

`c_seq_color`: specifies the color for a pattern.
Length: 5
Format: `0x2424001B colorcode`

This value specifies an index into the *Seq66* color palette. This feature is useful for distinguishing patterns more quickly in the pattern and song editors. Up to 32 colors (0 to 31) can be specified. See section 16 "Palettes for Coloring" on page 199. This section is saved only if a color has been specified. The lack of a color is given by a sequence color value of `c_seq_color_none` (-1).

25.2.24 SeqSpec c_seq_edit_mode

`c_seq_edit_mode`: specifies the editing mode (normal vs. drum) of a pattern.
Length: Indeterminate
Format: `0x2424001C`

This would specify that a pattern should be edited in drum mote (1) or normal mode (0), a *Kepler34* feature. However, in *Seq66* it is not implemented, nor read nor written. It is also better determined by the transposable status of a pattern, which the user can change with a button click.

25.2.25 SeqSpec c_seq_loopcount

`c_seq_loopcount`: specifies the number of times a pattern should play.
Length: 6
Format: `0x2424001D count:S`

This item is a feature to provide a user's request for a one-shot pattern, a pattern that plays once (or a few times) and never plays again. A count value of 0 would yield the normal behavior, playing endlessly during **Live** mode. Any other value would repeat the pattern the specified number of times. A loop-count is used in the one-shot step-edit feature used in recording a device's drum pattern. See section 6.8 "Pattern Editor / Bottom Row" on page 94. Note that setting a non-zero loop-count for a pattern is *not compatible* with playing the tune in **Song** mode.

25.3 MIDI Information

This section provides some useful, basic information about MIDI data. It can be helpful in troubleshooting. We tend to use the `hexer` tool for examine and troublesome MIDI file byte-by-byte.

25.3.1 MIDI Variable-Length Value

A *variable-length value* (VLV) is a quantity that uses additional bytes and continuation bits to encode large numbers. See https://en.wikipedia.org/wiki/Variable-length_quantity. The length of a VLV depends on the value it represents. Here is a list of the numbers that can be represented by a VLV:

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

See the functions `varinum_size()`, `write_varinum()`, and `read_varinum()`.

25.3.2 MIDI Track Chunk

Track chunk: `MTrk + length + track_event [+ track_event ...]`

- `MTrk` is 4 bytes representing the literal string "MTrk". This marks the beginning of a track.
- `length` is 4 bytes the number of bytes in the track chunk following this number. That is, the marker and length are not counted in the length value.
- `track_event` denotes a sequenced track event; usually there are many track events in a track. However, some of the events may simply be informational, and not modify the audio output, especially in the first track of an SMF 1 file.

A track event consists of a delta-time since the last event, and one of three types of events.

```
track_event = v_time + midi_event | meta_event | sysex_event
```

- `v_time` is the VLV for the elapsed time (delta time) from the previous event to this event.
- `midi_event` is any MIDI channel message such as Note-On or Note-Off.
- `meta_event` is an SMF meta event.
- `sysex_event` is an SMF system exclusive event.

25.3.3 MIDI System Events

For now, we simply summarize the System events. *Seq66* can read, insert, and write them. More to come.

1. `F0 ...`: System Exclusive.
2. `F1 ...`: Quarter Frame.
3. `F2 ...`: Song Position.
4. `F3 ...`: Song Select.
5. `F4 ...`: Undefined.
6. `F5 ...`: Undefined.
7. `F6 ...`: Tune Select.
8. `F7 ...`: SysEx Continue.
9. `F8 ...`: Clock Event.
10. `F9 ...`: Undefined.
11. `FA ...`: Start.
12. `FB ...`: Continue.
13. `FC ...`: Stop.
14. `FD ...`: Undefined.
15. `FE ...`: Active Sensing Message.
16. `FF ...`: Meta Message.

See section 8.3 "Event Editor / Edit Fields" on page 112; it describes some of the handling of system events.

25.3.4 MIDI Meta Events

Meta events are non-MIDI data of various sorts consisting of a fixed prefix, an event type, a length field, and the event data. *Seq66* tries to load, store, and write most meta events. Meta events are never sent to a device.

```
meta_event = 0xFF + meta_type + v_length + event_data_bytes
```

- `meta_type` is 1 byte, expressing one of the meta event types shown in the table that follows this list.
- `v_length` is length of meta event data, a variable length value.
- `event_data_bytes` is the actual event data.

Unfortunately, currently the processing of meta events is split between the `seq66::midifile` and `seq66::midi_vector_base`. "R/W" indicates both "Read" and "Written".

Here, we summarize the MIDI meta events data.

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 through 0F len text`: Other kinds of text events.

Table 17: MIDI Meta Event Types

Type	**Event**	**Seq66 Handling**
0x00	Sequence number	R/W
0x01	Text event	R/W
0x02	Copyright notice	R/W
0x03	Sequence or track name	R/W
0x04	Instrument name	R/W
0x05	Lyric text	R/W
0x06	Marker text	R/W
0x07	Cue point	R/W
0x08-0x0F	Other text events	R/W
0x20	MIDI channel (deprecated)	R only
0x21	MIDI port (deprecated)	R only
0x2F	End of track	R/W
0x51	Tempo setting	R/W and SeqSpec
0x54	SMPTE offset	R only
0x58	Time Signature	R/W c_timesig/c_perf_bp_mes/c_perf_bw
0x59	Key Signature	R/W
0x7F	Sequencer-Specific event	Seq66 data handled

10. `FF 2F 00`: End of Track.
11. `FF 51 03 tttttt`: Set Tempo, us/qn.
12. `FF 54 05 hr mn se fr ff`: SMPTE Offset.
13. `FF 58 04 nn dd cc bb`: Time Signature.
14. `FF 59 02 sf mi`: Key Signature.
15. `FF 7F len data`: Sequencer-Specific.
16. `FF F0 len data F7`: System-Exclusive

We need to make sure we read, save, and restore the items above that are marked as "Skipped" or just "Read", even if *Seq66* doesn't use them. Some are deprecated in the MIDI standard, and *Seq66* encodes them in a `SeqSpec`.

Windows MP in this application table is the built-in *Windows Media Player*. 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) (Seq66 only)
- Time Signature (0x58) (Seq66 only)
- Sequencer-Specific (0x7F) (Handled differently in Seq66)
- System Exclusive (0xF0) Sort of handled, functionality incomplete.

Also, all the text events should be handled by *Seq66*, but a lot more testing is needed.

Table 18: Application Support for Seq66 MIDI Files

Application	New	Original File
ardour	TBD	TBD
composite	TBD	TBD
gsequencer	No	No
lmms	Yes	Yes
midi2ly	Yes	TBD
midicvt	Yes	Yes
midish	TBD	TBD
muse	TBD	TBD
playmidi	TBD	TBD
pmidi	TBD	TBD
qtractor	Yes	Yes
rosegarden	Yes	Yes
superlooper	TBD	TBD
timidity	Yes	Yes
Windows MP	No	TBD

25.3.5 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.

25.3.6 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.

25.3.7 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.

25.3.8 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 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 in the tempo resolution allows a four-minute piece at 120 beats per minute to be accurate within 500 usec at the end of the piece.

25.3.9 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 denominator is a negative power of two: 2 represents a quarter-note, 3 represents an eighth-note, etc. The `cc` parameter expresses the number of MIDI clocks in a metronome click. The `bb` parameter expresses the number of notated 32nd-notes in a MIDI quarter- note (24 MIDI Clocks). Example:

```
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) indidicates 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.

If a time signature event is not present in a MIDI sequence, a 4/4 signature is assumed. In *Seq66*, 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.

25.3.10 SysEx Event (0xF0)

If the meta event status value is 0xF0, it is called a "System-exclusive", or "SysEx" event. *Seq66* does store these messages, but there are complications to be dealt with.

SysEx messages as sent and received are of the format:

```
F0 ID sysex data bytes F7
```

The ID is a manufacturer's ID. Normally a single byte, additional IDs can be represented by the sequence `00 xx yy`, to allow 16384 additional manufacturer IDs. Currently *Seq66* does not try to interpret the manufacturer ID.

There are three additional special ID codes that are *not* manufacturer-specific:

- **0x7D**. Used for private non-commercial purposes only. It has no standard format.
- **0x7E**. Represents a non-realtime system exclusive message. All MIDI devices can respond to it, although not immediately.
- **0x7F**. Represents a non-realtime system exclusive message. All MIDI devices can respond to it, immediately.

For more information on these events, see section 25.3.11 "Universal SysEx Events (0xF0 0x7E and 0xF0 0x7F)" on page 268.

Now, when encoded for storage in a MIDI file, the format of the bare message (first line) is padded so that it starts with a delta time, and includes the length (a variable-length quantity) of the SysEx including the final F7, but excluding the F0 and the length bytes.:

```
F0 ID sysex data bytes F7              (message)
delta F0 len ID sysex data bytes F7    (encoded)
```

SysEx events can be encoded in the following ways:

- **Single SysEx Message**.
- **Continuation Events**.
- **Escape Sequence**.

25.3.10.1 Single SysEx Message

A SysEx message that is *sent to a device* is not quite the same as a SysEx message that is *encoded in a MIDI file*. SysEx messages encoded in a MIDI file are preceded by a variable-length delta time, the byte F0, a variable-length length value, and the message, and a terminating F7, which *must* be present, and is counted in the length value. In the following example, the first line is the actual SysEx message (a General MIDI Enable message), and the second line is its encoding in a MIDI file, ignoring the preceding delta time:

```
F0 7E 00 09 01 F7          (message)
F0 05 7E 00 09 01 F7       (encoded)
```

25.3.10.2 Continuation Events

Some equipment needs the SysEx to be split into smaller chunks for processing. This could be accomplished with a number of smaller single-SysEx messages, but some manufacturers treat SysEx as if it supported running status (it does not). So the first message starts with F0, the next messages have only the SysEx data, and the last message ends with F7. However, when encoding in the MIDI file, each sub-packet begins with an F7. In between are the delta times to use to delay the sub-packets when sending to a slow device. An unencoded message with 3 packets is shown, with a bar for a separator for this discussion:

```
F0 43 12 00 | 43 12 00 43 12 00 | 43 12 00 F7
```

Encoded with a 200-tick delta time (`81 48`) between each message, the F7 being used as a *continuation* byte, with length values:

```
00 F0 03 43 12 00
81 48 F7 06 43 12 00 43 12 00
81 48 F7 04 43 12 00 F7
```

Since the first message is not terminated by F7 within the specified length, the next two F7s indicate a continuation. Note the F7 at the beginning of packets after the first one, and the F7 at the end of the last one.

25.3.10.3 Escape Sequence

An escape sequence is not SysEx, but it does use the F7 byte. It is used for encoding arbitrary bytes for messages such as Song Select. The first line shows such a message, and the second how it would be encoded.

```
F3 01
F7 02 F3 01
```

The format is `F7 len <all bytes to be transmitted>`.

So how to figure out what F7 means? Its interpretation is as follows:

- An F0 without a terminal F7 within the specified length is a Casio-style multi-packet message, and a flag (call it `multi`) should be raised to indicate this status.
- If F7 is encountered while `multi` is set, it is a continuation.
- If this continuation ends with F7, it is the last packet and `multi` should be cleared.
- If F7 is encountered while `multi` is *not* set, then the event is an escape sequence.

25.3.11 Universal SysEx Events (0xF0 0x7E and 0xF0 0x7F)

As noted earlier, 0x7E and 0x7F provide for non-realtime and realtime SysEx message. Immediately following these value is the "channel", which can be a manufacturer's ID or any value ranging from 0x00 to 0x7F (which is a wild-card for "all devices").

```
F7 7E channel subid1 subid2 data bytes F7      (non-realtime)
F7 7F channel subid1 subid2 data bytes F7      (realtime)
```

If `channel` is a device ID. If its value is F7, this is a global broadcast that all devices should heed. The `subid1` and `subid2` could be something like "01 00" for a long-form (full frame) time-code message.

A detailed description is beyond the scope of this document. Some messages supported by these messages are the *MIDI Master Volume*, *MIDI Full Frame*, and the *General MIDI System Enable/Disable* messages.

An interesting subset of these messages is *MIDI Show Control* (MSC):

```
F7 7F deviceid 02 commandformat command data F7      (MSC)
```

See [21]. Some simple controls can be included in the 'ctrl' file's set of macros.

25.3.11.1 Seq66 SysEx Handling

Seq66 is slowly gaining support for reading, storing, and sending SysEx events within a sequence. *Seq66* 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 handled as well.

The format of a bare (i.e. not encoded in a MIDI file) SysEx message is like the following, complex case:

```
F0 manid devid modelid direction address   data checksum terminator
F0 0x41  0x10  0x42    0x12      0x40007F  0x00 0x41     0xF7
```

The "manid" is a manufacturer's identifier. 0x41 is Roland, 0x24 is Hohner. 0x00 is used for adding two more codes to greatly expand the ID list. 0x7E is used to denote a non-realtime message, and 0x7F denotes a realtime message. The "devid" is the device identifier (e.g. 0x10 for Roland devices). The "devid" indicates which devices will accept the SysEx message. The "modelid" is the model identifier (0x42 for most GS synths). The "direction" is 0x12 for sending information or 0x11 for requesting information. The "address" is a 3-byte value on which the SysEx message should act. Devices provide an address map to define what is at each address. For example, this address might define a "GS Reset", which performs an initialization. The "data" is either the data to send (one or more bytes), or the size of the data we are requesting. Some devices will include a "checksum" for data integrity. All SysEx messages end with an F7 byte. But see section 25.3.10 "SysEx Event (0xF0)" on page 266.

25.3.12 Non-Specific End of Sequence

Any other statuses are deemed unsupportable in *Seq66*, and abort parsing with an error. If the `--bus` option is in force, it overrides the buss number (if any) stored with the sequence. This option is useful for testing a setup. Note that it also applies to new sequences. At the end, *Seq66* adds the sequence to the encoded tune.

25.4 More MIDI Information

This section goes into even more detail about the MIDI format, especially as it applies to the processing done by *Seq66*. The following sub-sections describe how *Seq66* parses a MIDI file.

25.4.1 MIDI File Header, MThd

The first thing in a MIDI file is The data of the header:

```
Header ID:        "MThd"         0x00: 4 bytes
MThd length:      6              0x04: 4 bytes
Format:           0, 1, or 2     0x08: 2 bytes (format 2 is rare)
No. of tracks:    1 or more      0x0a: 2 bytes
PPQN:             192            0x0c: 2 bytes (bit 15 = 0)
```

The header ID and its length are always the values above, "MThd" and 6. The formats that Seq66 supports are 0 or 1. SMF 0 has only one track, while SMF 1 can support an arbitary number of tracks. SMF 2 is rarely used, and *Seq66* does not support it. The next value is the number of tracks, 1 or more. The last value in the header is the PPQN value, which specifies the "pulses per quarter note", the basic time-resolution of events in the MIDI file. Common values are 96 or 192, but higher values are also common. (The highest possible value is 0x7FFF = 32767 but the MIDI functional limit is 31250, and anyway, *Seq66* limits it to 19200 for performance reasons.) *Seq66* and

its precursor, *Seq24*, default to PPQN = 192. For *Seq66*, this can be changed in the 'rc' file. Any MIDI file can also be rescale to another PPQN, and saved.

25.4.2 MIDI Track, MTrk

The next part of the MIDI file consists of the tracks specified in the file. 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"         4 bytes
Track length:  varies         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. In SMF 1 format, each track is assumed to cover a different MIDI channel, but the same MIDI buss. The MIDI buss *is* an important data item in the sequencer-specific section of *Seq66* MIDI files, however.

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 values 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.

```
Delta time:    varies         1 or more bytes
```

The running-time accumulator is incremented by the delta-time. The current time is adjusted as per the PPQN ratio, if needed, and stored with each event that is read from the MIDI file.

25.4.3 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         1 byte
```

If the high bit is set, it is a status byte. What does the status mean? To find out, 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         1 byte
Data byte 1:   varies         1 byte
```

If the status is a Note-On event, with velocity = data[1] = 0, then it is converted to a Note-Off event, a fix for the output quirks of some MIDI devices. If it is a 1-data-btye event (Program Change or Channel Pressure), then only data byte 0 is read. The one or two data bytes are added to the

event, the event is added to the current sequence. When the event is played, and the MIDI channel of the sequence is used, unless the MIDI channel for the sequence is "Any". In that case, the event's channel is used.

25.4.4 Meta Events Revisited

If the event status masks off to 0xF0 (0xF0 to 0xFF), then it is a Meta event. If the Meta event byte is 0xF7, it is called a "Sequencer-specific", or "SeqSpec" event. For this kind of event, then one to three manufacturer ID bytes and the length of the event are read.

```
Meta type:     varies         1 byte
Meta length:   varies         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 *Seq66*-specific events. These events are tagged with various values that mask off to 0x242400nn. The parser reads the tag:

```
Prop tag:     0x242400nn      4 bytes
```

These tags provide a way to save and recover *Seq24/Seq66* properties from the MIDI file: MIDI buss, MIDI channel, time signature, sequence triggers, and 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 marks the length (in MIDI pulses) of the sequence. 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 as the sequence name.

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

```
Seq number:    varies         2 bytes
```

Note that the sequence number might be modified latter to account for the current *Seq24* screenset in force for a file import operation. Any other `SeqSpec` type is simply skipped by reading the "length" number of bytes. The remaining sections simply describe MIDI meta events in more detail, for reference.

26 Kudos

This section gives some credit where credit is due. We have contributors to acknowledge, and have not caught up with all the people who have helped this project:

- *Tim Deagan (tdeagan)*: Fixes to the mute-group support.
- *0rel*: An important fix to add and relink notes after a paste action in the pattern editor.
- *arnaud-jacquemin*: A bug report and fix for a regression in mute-groups support. Also suggestions for enhancing mute-group support.
- *Stan Preston (stazed)*: Ideas for many improvements based on his *seq32* project. A lot of ideas. And a lot of code!
- *Animtim*: A number of bug reports and a new logo for *Sequencer64*.
- *jean-emmanuel*: Scrollable main-window support, other features and reports.
- *Olivier Humbert (trebmuh)*: French translation for the desktop files.
- *Oli Kester*: The creator of *Kepler34*, from which we got many clues on porting the user-interface to Qt 5 and Windows.
- *Tripayou*: Sylvain helped improve support for the AZERTY French keyboard in the 'ctrl' file.
- *Winko Erades van den Berg*: Added the Phrygian scale to the scales module.
- *phuel*: Provided an upgrade to check-mark the selected buss and channel in a pattern's pop-up menu.
- *Milkii Brewster*: Always pointing out issues and upgrades, and publicizing Seq66 on various "bulletin boards".
- *All bug reporters*. Couldn't do this well withou y'all!

Additional bug-reporters and testers:

- *F0rth*: A request for scripting support, a possible future feature.
- *gimmeapill*: Testing, bug-reports, and, um, "marketing".
- *georgkrause*: A number of helpful bug reports.
- *goguetchapuisb*: Found that *Sequencer64* native JACK did not properly handle the copious Active Sensing messages emitted by Yamaha keyboards.
- *milkmiruku*: Mainwids issues and many ideas, suggestions, feature requests, and bug report. An endless source of suggestions!
- *muranyia*: Feature request for numbered piano keys and bug-reports.
- *simonvanderveldt*: Issues with window sizing and more.
- *ssj71*: A request for an LV2 plugin version, a possible future feature.
- *triss*: A request for OSC support, a possible future feature. We added some OSC support in order to play well with the *Non Session Manager* (*NSM*).
- *layk*: Some bug reports, and, we are pretty sure, some nice videos that demonstrate *Seq66* on *YouTube* [10].
- *matt-bel*: Reported a regression from *Seq24*, which could use a MIDI control event to mute / unmute multiple patterns at once, a cool feature!
- *zigmhount*: A pending request for a control that would automatically set up a pattern for recording and playback with one "click".
- *grammoboy* and *J. Liles*: Bug reports and other help with *NSM* support.
- *Houston4444*: Similarly, help with *RaySession*, a work-alike of *NSM*, written in *Python*.
- *unfa*: Bug reports for coloring, and for inspiring the "*.palette" file feature, as well as making coloring more comprehensive.
- *slightlynasty*: Ideas for updating the song-record functionality and improving the handling of large numbers of MIDI ports. A lot to think about.
- *grammoboy2*: Lots of suggestions to make sure that *Seq66* is absolutely compliant with the API of *Non Session Manager*.

... and there are many more to add to this list....

There are a number of authors of *Seq24*. ideas from other *Seq24* fans, and some deep history. All of these people, and more, have contributed to *Seq66*, whether they know it or not. The original author is Rob C. Buse. Without his work, we would never have started this years-long project:

> *Seq24* is a real-time MIDI sequencer. It was created to provide a very simple interface for editing and playing MIDI 'loops'. After searching for a software based sequencer that would provide the functionality needed for a live performance, there was little found in the software realm. I set out to create a very minimal sequencer that excludes the bloated features of the large software sequencers, and includes a small subset of features that I have found usable in performing. Written by Rob C. Buse. I wrote this program to fill a hole. I figure it would be a waste if I was the only one using it. So, I released it under the GPL.

Taking advantage of Rob's generosity, we've created a reboot, a refactoring, an improvement of *Seq24*. It preserves (somewhat) the lean nature of *Seq24*, while adding many useful features. Without *Seq24* and its authors, *Seq66* would never have come into being.

27 Summary

Contact: If you have ideas about *Seq66* or a bug report, please email us (at mailto:ahlstromcj@gmail.com). If it's a bug report, please add **[BUG]** to the Subject, or use the GitHub bug-reporting interface.

28 References

The *Seq66* reference list.

References

[1] ALSA team. *Advanced Linux Sound Architecture (ALSA) project homepage*. http://www.alsa-project.org/. ALSA tools through version 1.0.29. 2015.

[2] superuser.com. *How do I increase the number of MIDI through ports in ALSA?* https://superuser.com/questions/973973/how-do-i-increase-the-number-of-midi-through-ports-in-alsa 2015.

[3] laborejo. *Agordejo Session Manager, part of the Laborejo Software Suite* https://www.laborejo.org/agordejo/, http://git.laborejo.org/lss/agordejo.git 2021.

[4] Seq66 Arch Linux User Repository. *Seq66 AUR PACKAGE file for version 0.99.4* https://aur.archlinux.org/packages/seq66-git/ 2022.

[5] Jay Capela Music. *"Combine": A Seq24 Demonstration*. https://www.youtube.com/watch?v=fUiXbVTObJQ. 2010.

[6] Various. *Forks of Seq66*. https://github.com/sivecj/seq66. 2021.

[7] JACK team. *JACK Audio Connection Kit*. http://jackaudio.org/. 2015.

[8] Oli Kester. *Kepler34: Seq24 for the 2010s*. https://github.com/oli-kester/kepler34. 2010-2016.

[9] Linux Audio Users. *Unofficial Linux audio users IRC channel*. http://kiwiirc.com/nextclient/#irc://irc.freenode.net/#lau. 2021.

[10] Lassi Ylikojola. *Many demo videos of Sequencer64*. https://www.youtube.com/watch?v=YStYVjFv1TM, https://www.youtube.com/watch?v=GBlEP8Ffqss, https://www.youtube.com/watch?v=4gG8SvJxJkA&t=28s, https://www.youtube.com/watch?v=n4Z4WPK6FpA. 2010-2017.

[11] Author unknown. *A description of Seq66*. https://libreav.org/software/seq66. 2019.

[12] Tobias Erichsen. *Private stuff & software for audio, midi and more* https://www.tobias-erichsen.de/software.html. 2022.

[13] Phil Kerr. *The Linux MIDI-HOWTO*. https://tldp.org/HOWTO/MIDI-HOWTO.html. 2002.

[14] Murks. *ALSA and JACK MIDI explained (by a dummy for dummies)*. https://freeshell.de/~murks/posts/ALSA_and_JACK_MIDI_explained_(by_a_dummy_for_dummies)/. 2010.

[15] Coolsoft. *Coolsoft MIDIMapper (Windows)*. https://coolsoft.altervista.org/en/midimapper. 2018.

[16] Jamie O'Connell and Jerry Jorgenrud. *MIDI OX and other related software (Windows)*. http://www.midiox.com/. 2017.

[17] Coolsoft. *Coolsoft VirtualMIDISynth (Windows)*. https://coolsoft.altervista.org/en/virtualmidisynth. 2018.

[18] linuxaudio.org. *seq24: toggle sequences with a MIDI controller*. http://wiki.linuxaudio.org/wiki/seq24togglemiditutorial. 2013.

[19] midi.org. *Summary of MIDI Messages*. https://www.midi.org/specifications/item/table-1-summary-of-midi-message#2. Year unknown.

[20] Chris Ahlstrom. *Extension of midicomp/midi2text to convert between MIDI and ASCII text format*. https://github.com/ahlstromcj/midicvt. 2015-2016.

[21] *Digital Sound and Music: MIDI Show Control*. https://digitalsoundandmusic.com/6-2-5-non-musical-applications-for-midi/. 2023.

[22] Roy Vegard. *Configurator software for the Korg nanoSERIES of MIDI controllers*. https://github.com/royvegard/Nano-Basket. Warning: this code may require an earlier version of Python than present in one's Linux distro or in Windows. 2015.

[23] JACK Audio. *New Session Manager, an offshot of Non Session Manager*. https://github.com/jackaudio/new-session-manager 2021.

[24] Stephen Sinclair *Open Sound Control (library)*. https://github.com/radarsat1/liblo 2019.

[25] Simon W. Fielding. *QjackCtl and the Patchbay.* https://www.rncbc.org/drupal/node/76. 2008.

[26] sferamusic *HOW-TO: DIY hybrid Sequencer / MIDI USB Hub ....* https://www.rncbc.org/drupal/node/7://steemit.com/music/@sferamusic/how-to-diy-hybrid-sequencer-midi-usb-hub-aka-piseq-using-raspberry-pi-and-external-midi-cor 2017.

[27] PortMedia team. *Platform Independent Library for MIDI I/O.* http://portmedia.sourceforge.net/. 2010.

[28] Benjamin Robert Tubb *MIDI PPQN Duration Calculator.* https://demonstrations.wolfram.com/MIDIPPQNDurationCalculator/. 2011.

[29] Dmitry Marakasov <amdmi3@amdmi3.ru> (Repology). *Known Seq66 Repositories.* https://repology.org/project/seq66/versions. 2023.

[30] *QjackCtl.* Rui Nuno Capelo (rncbc). https://sourceforge.net/projects/qjackctl/. https://qjackctl.sourceforge.io/qjackctl-index.html. https://git.code.sf.net/p/qjackctl/code qjackctl-git. 2021.

[31] Houston4444. *RaySession Session Manager* https://raysession.tuxfamily.org/en/ https://github.com/Houston4444/RaySession 2021.

[32] Gary P. Scavone. *The RtMIDI Tutorial.* https://www.music.mcgill.ca/~gary/rtmidi/. 2016.

[33] Seq24 Team. *The home site for the Seq24 looping sequencer.* http://www.filter24.org/seq24/download.html. 2010.

[34] Seq24 Team. *The home site for the Seq66 looping sequencer.* https://edge/launchpad.net/seq24. 2016.

[35] Excds. *A simple mapping for toggling the LEDs on the Novation launchpad together with seq24.* https://github.com/Excds/seq24-launchpad-mapper. 2013.

[36] Stan Preston (stazed). *The home site for the Seq32 looping sequencer.* https://github.com/Stazed/seq32. 2016.

[37] Chris Ahlstrom. *A reboot of the Seq24 project as "Seq66".* https://github.com/ahlstromcj/seq66/. 2015-2021.

[38] Unknown. *Timing accuracy and resolution.* http://midi.teragonaudio.com/tutr/seqtime.htm. 2000?.

[39] Timidity++ Team. *Download site for Timidity++ source code.* http://sourceforge.net/projects/timidity/. 2015.

[40] VMPK Team. *Virtual MIDI Piano Keyboard.* http://vmpk.sourceforge.net/. 2015.

[41] Donya Quick *Working with MIDI on Windows* http://donyaquick.com/midi-on-windows/#x1-120004.1. 2021.

[42] The Wootangent man. *Seq24 Tutorial Videos, Part 1 and Part 2.* http://wootangent.net/2010/10/linux-music-tutorial-seq24-part-1/. http://wootangent.net/2010/10/linux-music-tutorial-seq24-part-2/. 2010.

[43] Various. *Links to other usages of the seq** applications.* https://www.youtube.com/watch?v=LjHPtlvjgnM AndrewsVintageKeyboards: Yamaha TG100 with seq66. https://www.youtube.com/watch?v=ESHKiHx09AA&t=78s AndrewsVintageKeyboards: Yamaha TG100 with sequencer66. https://www.youtube.com/watch?v=_r7qXOo7kgg Julian L: Seq64 demo. https://www.youtube.com/watch?v=VCga2X4lJfg Julian L: Seq64 demo 2.

Index

–auto-ports, 149
–bus option, 22
–session-tag, 127
-0, 140
-A, 138
-C, 139
-F, 140
-H, 140
-J, 139
-K, 139
-L, 140
-M, 139
-N, 139
-R, 138
-S, 139, 140
-T, 138
-U, 140
-V, 137
-W, 139
-X, 138
-a, 138
-b, 138
-c, 140
-f, 140
-g, 139
-h, 137
-j, 139
-k, 139
-l, 138
-m, 138
-n, 138
-o, 140
-p, 139
-q, 138
-r, 138
-s, 138
-t, 139
-u, 140
-v, 137
-w, 139
--alsa, 138
--auto-ports, 138
--bus [buss], 138
--client-name, 138
--config basename, 140
--help, 137
--hide-ports, 138
--home [directory], 140
--inverse, 139
--jack-connect, 139
--jack-master, 139
--jack-master-cond, 139
--jack-midi, 139
--jack-session-uuid [uuid], 140
--jack-slave, 139
--jack-start-mode [x], 139
--jack-transport, 139
--locale localename, 140
--manual-ports, 138
--no-jack-connect, 139
--no-jack-midi, 139
--no-jack-transport, 139
--no-nsm, 138
--nsm, 138
--option opvalue, 140
--playlist [filename], 138
--ppqn [ppqn], 138
--priority, 139
--rc filename, 140
--reveal-ports, 138
--session-tag section, 140
--show-keys, 139
--show-midi, 138
--smf-0, 140
--user-save, 140
--usr filename, 140
--verbose, 137
--version, 137
[allow-click-edit], 152
[auto-option-save], 152
[jack-transport], 150
[last-used-dir], 152
[loop-control], 164
[manual-ports], 149
[midi-clock-mod-ticks], 147
[midi-clock], 146
[midi-control-file], 142
[midi-input], 148
[midi-meta-events], 148
[playlist], 153

[recent-files], 153
[reveal-ports], 150
[snap-split], 152
[user-instrument-definitions], 157
[user-instrument-n], 157
[user-interface-settings], 158
[user-midi-bus-definitions], 156
[user-midi-bus-n], 156
[user-midi-settings], 160
[user-options], 161

Add, 190
Add Song, 190
Add Time Signature Event, 77
alt-left-click solo, 69
Apply Song Transpose, 36
armed, 247
Armed/Mute Toggle, 97
arrow keys, 187
auto-connect, 222
auto-note, 87
auto-shift, 196, 203, 204
auto-step, 99, 248
    recording, 99
AZERTY, 177, 240

background recorder, 51
Background Sequence, 83
bank, 248
Beat, 86
Beat Width, 77
Beat width, 51
beat width, 77
Beats Per Bar, 77
beats per bar, 77
Beats/bar, 50
Bold Slot Font, 46
BPM, 72
bpm
    page increment, 170
    step increment, 168
bugs
    event delete key, 115
    event insert key, 115
bus, 248
    naming, 46
buss, 248
    mapping, 39
    naming, 46
    override, 39, 160
Buss Name, 39
buss names
    long, 46
    short, 46

Category, 112
Channel, 51, 114
Channel Number, 111
channel split, 251
Chord Generation, 83, 91
chord generation, 78, 91
Chord Types, 78
Clear, 115
Clear (maps), 40
Clear Mute Groups, 36
Clear Set, 193
client ID, 41
Client Name:ID/UUID, 41
Client Number, 39
client number, 39
Client:port buss names, 46
Clock Start Modulo (ticks), 40
Configuration Files, 54
Connect, 48
Copy Current Set, 36
Copy Pattern, 62
copy set, 36
Copy/Paste, 89
count-in recording, 70
Create (maps), 40
Create File, 190
ctrl-left-click new pattern, 69
Cut Pattern, 62

D0, 114
D1, 114
daemonize, 140
data
    grab handle, 91
Data Bytes, 111
data pane, 91
default PPQN, 159
Default Seq66 PPQN..., 49
Delete, 190
Delete Pattern, 62
Delete Song, 190

deprecated, 139
Disable, 204
Disconnect, 48
Double click for pattern editor, 46
double-click
    pattern slot, 46
down arrow, 88
draw mode, 86
Drum Note Mode, 91
Dump, 115

edit
    clear mute groups, 36
    configuration, 54
    copy set, 36
    default ppqn, 49
    load mute groups, 36
    mute all tracks, 36
    paste set, 36
    pattern, 52
    preferences, 36
    resume notes, 49
    session, 54
    sets-mode, 49
    song editor, 36
    song transpose, 36
    toggle all, 36
    unmute all tracks, 36
    use file ppqn, 49
    UUID, 54
Edit Events In Tab, 62
Edit Pattern In Tab, 62
Edit Pattern In Window, 62
editing shortcut, 62
Editor Key Height, 45
empty pattern, 64
empty slot double-click, 62
Enable, 204
Event, 113
event
    compression, 88
    drum note, 59
    note, 59
    select, 87
    stretch, 88
    tempo, 59
Event Category, 112
Event Category Selector, 94
event data editor
    mouse wheel, 94
event edit, 203
event editor
    channel, 114
    channel number, 111
    clear events, 115
    data byte 1, 114
    data byte 2, 114
    data bytes, 111
    dump events, 115
    event category, 112
    event name, 111, 113
    event time, 113
    event timestamp, 112
    hex data, 114
    index number, 110
    insert event, 115
    links, 111
    modify event, 115
    save events, 115
    text data, 114
    tick time, 114
    time stamp, 110
Event Name, 111
Event Selection, 87, 95
event strip, 90
events
    insert, 90
Existing Event Menu, 95
expand, 71, 78, 97
Expand Song Roll, 104
export, 249
exportable, 130
External Live Frame for Set, 61
External live frame for set 0, 59

fast forward, 205
file PPQN, 159
fingerprint, 162
Fingerprint Size, 45
fingerprint size, 45
follow jack, 204
Follow Progress, 81, 103
font
    bold, 46
full paths, 46

Global background/scale/key, 46

global pattern setting
background, 46
key, 46
scale, 46
global-sequence, 82
globals
background etc., 46
grid
bold, 46
swap coordinates, 46
grid modes, 171
Grid scaling & spacing, 45
Grid Snap, 81
Grid/Record Snap, 103
group, 249
learn, 204
off, 204
on, 204
Group Patterns, 197
group-learn, 204

Help
about, 55
app keys, 56
build info, 55
song summary, 55
tutorial, 56
user manual, 56
Hex, 114

Index Number, 39, 110
index number, 39
Input Bus, 63
Input Buses, 42
input buses, 42
input by buss, 43
input by channel, 43
Input Options, 43
input options, 43
Input/Output Virtual Ports, 43
Insert, 115

JACK
auto-connect, 48
live mode, 48
master conditional, 47
native midi, 48
song mode, 48
transport, 47
transport master, 47
jack
auto-connect, 222
midi, 220
reveal-ports, 150
transport, 220
jack server
settings, 48
JACK Server Settings, 48
JACK Start mode, 48
jack sync
connect, 48
disconnect, 48
start mode, 48
transport/midi, 47
JACK toggle, 204
jitter, 79

keep queue, 203, 249
Keep-Queue Status, 73
key height, 45, 162
Key of Sequence, 81
keyboard
disable, 204
enable, 204
group off, 204
group on, 204
igrave, 204
learn, 204
mute-group slots, 204
sequence toggle keys, 204
keys
-, 62
=, 62, 89
[, 58, 68
], 58, 68
0, 89, 100, 108
alt, 68
apostrophe, 73, 204
arrow, 184, 187
arrows, 85, 108
avoid ctrl/alt, 68
AZERTY, 150, 164, 216
backspace, 89, 108
blank, 165
c, 89
control, 165
copy, 108

ctrl-a, 79, 87
ctrl-arrows, 85
ctrl-c, 89, 108
ctrl-d, 89
ctrl-e, 87
ctrl-end, 85
ctrl-home, 85
ctrl-k, 83, 89
ctrl-n, 87
ctrl-v, 89, 108
ctrl-x, 89, 108
ctrl-z, 79
decrement set, 68
del, 89
delete, 107, 108
down-arrow, 108
Esc, 75, 87
esc (stop), 72, 89
event edit, 62
f, 89
focus, 202
gotchas, 203
hjkl, 76, 85, 108
Home, 27, 58
hot, 68
i, 75
increment set, 68
keep queue, 68, 203
l, 105, 169
o, 89
one-shot queue, 69
oneshot, 204
p, 75, 87, 90
page-down, 85, 100, 108
page-up, 85, 100, 108
paste, 108
pattern edit, 62
pattern shift, 68
pattern toggle, 68
period (pause), 89, 102
q, 89
qt, 202
queue, 68, 203
r, 89, 105
replace, 69
right ctrl, 68
screenset down, 58, 68
screenset play, 58
screenset up, 58, 68
semicolon, 73
shift, 203
shift page-down, 100
shift page-up, 100
shift-V, 108
shift-v, 89
shift-z, 89, 108
shortcut, 68
slash, 68
slot-shift, 203
snapshot, 68, 203
space (play), 72, 89
space (toggle play), 102
t, 89
u, 89
up-arrow, 108
V, 100
v, 89, 100, 108
vi, 76
x, 75, 87, 90
Z, 100
z, 89, 100, 108

L anchor, 105
L button, 27, 169
L marker, 26, 102, 105
L/R Collapse, 104
L/R Expand, 104
L/R Expand and Copy, 104
L/R Loop, 104
Learn, 204
left arrow, 88
left click, 105
Length Fraction, 51
LFO Panel, 95
Links, 111
linux
    system MIDI Thru, 148, 217
Lists Active, 190
live
    external, 59
Live Frame, 57
Live Grid, 57
Live Loop Count, 97
live mode, 48, 65
Load File, 198
Load Most Recent File (startup), 46

load most-recent, 46
Load Song, 190
Lock Main Window, 46
lock-main-window, 162
log, 140
Long Port/Buss Name, 46
Loop, 72
loop, 247
    grid modes, 70
loop mode, 26
Loop Record Type, 97

main window, 57
    lock, 46
marker
    mode, 105
    movement, 105
Measure, 86
measures ruler, 104
    left-click, 105
    right-click, 105
menu mode, 204
merge, 70, 97
Merge Into Pattern, 63
Meta Events, 40
Meta events, 34
meta text, 125
Metro Buss, 51
metronome, 69
    beat width, 51
    beats/bar, 50
    buss, 51
    channel, 51
    length fraction, 51
    note, 51
    patch, 51
    reload, 51
    velocity, 51
midi
    VLV, 261
midi clock
    buss name, 39
    client number, 39
    clock start modulo, 40
    index number, 39
    meta events, 40
    off, 40
    on (mod), 40
    on (pos), 40
    port disabled, 40
    port maps, 40
    port name, 39
    port number, 39
midi control
    input, 42
    input buss, 42
    output, 40
    output buss, 40
MIDI Control Input Bus, 42
MIDI Control Output Bus, 40
midi i/o
    port mapping, 40
    remove mapping, 40
MIDI I/O Port Maps, 40
MIDI Out Channel, 78
MIDI Out Device (Buss), 78
MIDI Record Quantized, 97
MIDI Record Toggle, 97
MIDI Thru, 148, 217
MIDI Thru Toggle, 97
mode
    draw, 86
    paint, 86
Modify, 115, 190
modify event-data, 91
modify pitch, 79
Modify Song, 190
mouse
    ctrl-left-click, 88
    ctrl-left-click-drag, 88
    left-click, 75, 88
    left-click-drag, 75, 88, 91
    right-click-drag, 91
Musical Scale, 82
musical scales, 82
mute all, 36
Mute All Tracks, 36
mute groups, 36
mute-group, 249
mute-group control, 166
Mute-group slots, 204
Mute-Groups, 197
Mute-Groups Table, 197
mutes
    clear all mutes, 198
    groups, 197

load file, 198
mutes file, 198
patterns, 197
read/write format, 198
save file, 198
table, 197
trigger mode, 198
Mutes File, 198

New, 59
new
allow-click-edit, 152
snap-split, 152
New Pattern, 61
new seqedit, 162
no-daemonize, 140
non-playback mode, 48
Note, 51
note
selection box, 88
Note Length, 81
Note Map, 91
note resume, 162
note step, 85
Notes, 86
notes
inserting, 87
nsm
new session, 121
old session, 121

Off, 40
On (Mod), 40
On (Pos), 40
one-shot, 71, 250
playback, 250
recording, 250
one-shot queue, 69
one-shot reset, 71
oneshot, 98, 204, 250
playback, 250
recording, 250
output
ports and clocks, 39
Output Bus, 63
Output Channel, 63
output ports, 39
overdub, 70, 97
overwrite, 71, 97

paint mode, 86, 107
palette, 55
palette, 139
Palette File Base Name, 55
pan
seqroll notes, 100
seqroll time, 100
Panic, 72
Paste Pattern, 62
paste set, 36
Paste To Current Set, 36
Patch, 51
Pattern, 52
pattern, 247
alt-left-click, 69
beat, 64, 105
BPM, 72
bus-channel, 64
buss, 63
buss-channel, 105
channel, 63, 105
color, 62
color menu, 59
contents, 64
copy, 62
ctrl-left-click, 69
cut, 62
delete, 62
edit in tab, 62
edit in window, 62
end marker, 86
events in tab, 62
external live frame, 61
free channel, 64
input buss, 63
keep-queue, 73
left click, 65, 69
left click-drag, 69
length, 64
loop, 72
merge, 63
middle click, 69
mute, 69
mute toggle, 69
name, 64, 105
new, 59, 61

panic, 72
paste, 62
Pause, 72
Play, 72
record toggle, 62
right click, 59, 60, 69
set name, 72
set number, 72
set reset, 72
shift-left-click, 69, 106
slot, 58
song record, 72
song/live, 73
split, 107
status, 64
stop, 72
tap tempo, 73
title, 105
trigger count, 105
unmute, 69
Pattern Color, 62
pattern edit, 203
pattern editor
background sequence, 83
beat width, 77
beats/bar, 77
chord generation, 83, 91
chord types, 78
copy, 89
copy/paste, 89
cut, 89
data to midi buss, 97
delete, 89
deselect notes, 88
drum mode, 91
event compression, 88
event selection, 95
event selector, 94
event stretch, 88
existing events, 95
grid snap, 81
key, 81
left click, 88
length, 77
LFO, 95
live loop count, 97
midi data pass-through, 97
midi out channel, 78
midi out device, 78
move notes in time, 88
name, 77
note length, 81
note map, 91
number, 77
paste, 89
progress bar, 78
quantize, 79
quantized record, 97
record midi data, 97
recording type, 97
redo, 79
scale, 82
select multiple notes, 88
select note, 88
time scroll, 86
time signature, 77
tools, 79, 81
transpose notes, 88
transpose toggle, 91
undo, 79
velocity, 99
vol, 99
zoom, 81
pattern extension, 109
Pattern Length, 77
pattern subsection, 107
pattern-editor
fix, 80
patterns column
ctrl-left-click, 106
double-click, 106
left-click, 106
Patterns Panel, 57
patterns panel
inverse muting, 69
solo, 69
patterns-panel
modes, 171
pause, 72, 203
performance, 100, 248
performance editor, 36
performance mode, 48
piano roll
beat, 86
measure, 86
notes, 86

virtual piano keyboard, 86
pipewire, 15
Play and Pause, 72
play screen, 193, 248
play-set, 22, 249
playback
one-shot, 250
playback mode, 48
playing set, 204
playlist
number, 186
playlist, 138
song-storage directory, 186
tag, 186
title, 186
playlist editor
add list, 190
add song, 190
create file, 190
delete list, 190
delete song, 190
lists active, 190
load song, 190
modify song, 190
save lists, 190
pointer position, 205
port
remove mapping, 40
mapping, 39, 40
override, 39, 160
Port Disabled, 40
port mapping, 40
port name, 39
Port Number, 39
port number, 39
port-naming, 141
long, 142
pair, 142
short, 141
ports
manual, 149, 223
virtual, 43, 149, 223
virtual auto-enable, 43
Ports and Clocking, 39
ppqn, 110
$ shortcut, 110
Preferences, 36
progress
note-max, 162
note-min, 162
progress bar
thick, 46
progress box
height, 162
shown, 162
width, 162
Progress Box Shown, 45
Progress Boxes, 45
progress-box shown, 45
progress-box size, 45
progress-note-max, 162
progress-note-min, 162
pulseaudio, 226
pulses, 249

qseq66.usr, 153
quantize, 79
Quantize Selection, 79
queue, 203
cancel, 66
clear, 67
end, 67
keep, 68, 203, 249
one-shot, 69
solo, 69
temporary, 68
quiet, 46
QWERTZ, 179, 240

R anchor, 105
R marker, 26, 105
randomize, 79
rc
auto-option-save, 140
jack-transport, 139, 140
manual-ports, 138
mute groups, 36
playlist, 138
reveal-ports, 138
Read/Write Format, 198
record
by bus, 43
by channel, 43
merge, 70
overdub, 70
thru buss, 51

Record Buss, 51
Record Toggle, 62
Record-by-Bus, 43
Record-by-Channel, 43
recorder
    buss, 51
    thru channel, 51
recording
    auto-step, 99
    expand, 71, 78, 97
    expand issue, 97
    merge, 97
    one-shot, 71, 250
    one-shot reset, 71
    oneshot, 98
    overdub, 97
    overwrite, 71, 97
    step-edit, 99
recording toggle, 89
Redo, 79, 103
Reload Metronome, 51
Reload Mute Groups, 36
remove mapping, 40
restart, 41
    automatic, 30, 35, 126, 129
    manual, 40, 125
Restart Seq66, 41
Resume Note Ons..., 49
rewind, 205
right arrow, 88
right click, 105

Save, 115
save background sequence, 83
Save Current Palette, 55
Save File, 198
Save Lists, 190
save musical key, 82
save musical scale, 83
scale identifier, 83
scaling, 141
screen-set, 58
screenset, 248
scroll
    ctrl scroll, 100
    horizontal pan, 100
    horizontal zoom, 100
    normal scroll, 100
    notes pan, 100
    notes zoom, 100
    shift scroll, 100
    timeline pan, 100
    timeline zoom, 100
    vertical pan, 100
    vertical zoom, 100
selection
    all, 87
    analyze, 89
    clear, 89
    deselect, 88
    edge fix, 89
    events by channel, 87
    notes by channel, 87
    quantize, 89
    randomize, 89
    relink notes, 89
    remove unlinked notes, 89
    repitch, 89
    tighten, 89
selection action, 88
SEQ66_SESSION_TAG, 128
SeqSpec, 33, 251, 252
    c_backsequence, 258
    c_bpmtag, 255
    c_gap_(ABCDEF) (unused), 257
    c_midibus, 252
    c_midichannel, 254
    c_midiclocks, 254
    c_midictrl (obsolete), 257
    c_midiinbus, 253
    c_musickey, 258
    c_musicscale, 258
    c_mutegroups, 256
    c_notes, 254
    c_perf_bp_mes, 259
    c_perf_bw, 259
    c_reserved_(2), 259
    c_seq_color, 260
    c_seq_edit_mode (Kepler34), 260
    c_seq_loopcount, 260
    c_tempo_map, 259
    c_tempo_track, 260
    c_timesig, 255
    c_transpose, 258
    c_trig_transpose, 256
    c_triggers, 254

c_triggers_ex, 256
table, 252
sequence, 247
sequence extension, 109
Sequence toggle keys, 204
Session, 54
session
new, 124
non, 124
session-tag, 127
sessions
/etc/hosts, 120
lash, 127
liblo library, 120
loopback interface, 120
non-starter, 120
nsm, 118
OSC, 118
signals, 116
ui, 124
Set, 72
Set 0, 194
Set List, 193
set master
clear set, 193
grid, 193
list, 193
number/name, 193
set 0, 194
set triggers, 194
show set info, 193
up/down buttons, 193
Set Name, 72
Set Number/Name Fields, 193
Set Reset, 72
set-size, 45
Sets Grid, 193
Sets Mode, 49
sets-mode, 141
additive, 141
allsets, 141
autoarm, 141
normal, 141
shift left click, 65, 105
shift-left-click live-frame, 69
shift-left-click solo, 106
Show Full Path of Recent Files in Menu, 46
Show Set Info, 193
slot, 247
double-click, 61
empty slot right-click, 59
slot-shift, 203
SMF 0, 251
snapshot, 203, 250
solo
true, 69
song, 248
info, 125
Song Editor, 36
song editor, 36
collapse, 104
deletion, 107
double-click, 107
draw, 107
dual, 101
expand, 104
expand and copy, 104
expand song roll, 104
follow, 103
grid/record snap, 103
handle, 106
horizontal zoom, 107
insert, 107
inverse muting, 106
left-click, 107
left-click-right-hold, 107
middle click, 107
middle-click, 107
multiple insert, 107
mute indicator, 105
muting, 106
note events, 106
paint, 103
pattern subsection, 107
play loop, 104
redo, 103
right left click, 107
right-click-hold, 107
right-left-hold-drag, 107
section deselection, 106
section expansion, 107
section length, 106
selection, 107
selection movement, 106
solo, 106
split pattern, 107

tempo events, 106
transpose, 104
trigger transpose, 104
undo, 103
vertical zoom, 107
zoom, 103
song mode, 48, 100, 204
Song Record, 72
song transpose, 36, 104
step, 85
step-edit, 99, 248
recording, 99
sticky options, 222
Stop, 72
style sheet, 143
Suppress startup error messages, 46
Swap Coordinates, 46

tap bpm, 205
Tap Tempo, 73
tempo events, 251
tempo-track-number, 40, 148
Text data, 114
thru, 97
Thru Buss, 51
Thru Channel, 51
Tick time, 114
tighten, 79
Time, 112, 113
Time Scroll, 86
time signature events, 251
Time Stamp, 110
todo
high precision events, 90
toggle all, 36
Toggle All Tracks, 36
toggle JACK, 204
toggle mutes, 205
Toggle Section Painting, 103
Toggle Song/Live Mode, 73
Tools, 79
Track Name, 77
Track Number, 77
Transport/MIDI, 47
Transpose, 91, 104
trigger, 248
Trigger Transpose, 104
Triggers, 194, 198

Undo, 79, 103
unmute all, 36
Unmute All Tracks, 36
up arrow, 88
Up/Down Buttons, 193
Use File's PPQN..., 49
usr
–user-save, 154
-u, 154
beat-width, 160
beats-per-bar, 160
beats-per-minute, 160
bpm-maximum, 91, 160
bpm-minimum, 91, 160
bpm-page-increment, 160
bpm-precision, 41, 160
bpm-step-increment, 160
buss-override, 160
convert-to-smf-1, 160
interface settings, 158
midi-ppqn, 159
option-browser, 161
option-daemonize, 161
option-logfile, 161
option-pdf-viewer, 161
page increment, 170
step increment, 168
user-instrument-definitions, 157
user-instrument-n, 157
user-interface-settings, 158
user-midi-bus-definitions, 138, 156
user-midi-bus-n, 156
user-midi-settings, 28, 138
user-session, 138
velocity-override, 160
usr config, 42
UUID, 54

variable-length value, 261
variset, 73, 141, 158
slash key, 68
Velocity, 51, 99
verbose, 46
Verbose Console Output, 46
virtual keyboard
right-click, 86
Virtual Piano Keyboard, 86
Virtual Ports Auto-Enable, 43

VMPK, 219
Vol, 99

warning
    down arrow, 88
    note loss, 88
    up arrow, 88
    wrap-around notes, 88
warnings
    usr config, 42
window
    close, 100
window scaling, 45
windows
    midi mapper, 211

Zeroes, 198
Zoom, 81
zoom, 107
    seqroll notes, 100
    seqroll time, 100
zoom keys, 89
Zoom Out and Zoom In, 103