Seq66 User Manual 0.97.0

Chris Ahlstrom
(ahlstromcj@gmail.com)

October 12, 2021

Seq66, Theme Dark-Cold Master, Fluxbox WaterDrops Style

Contents

List of Figures

List of Tables

1 Introduction

This document describes "Seq66", a reboot of *Seq24* and a rewrite of *Sequencer64*, through version 0.97.0. The following project supports *Seq66* and documentation:

- https://github.com/ahlstromcj/seq66.git

Seq66 is *Sequencer64* refactored for newer versions of *C++* with cruft cleanup. It drops the *Gtkmm* user-interface in favor of *Qt 5*, and has better handling of sets, mute-groups, sessions, configuration files, and more. It supports for the *Non Session Manager*, the ability to modify the color palette, and Qt style-sheets. Be prepared to note some significant differences between *Seq66* and *Sequencer64*.

We also have many contributors to acknowledge. Please see section 23 "Kudos" on page 177. It is out-of-date! If your name is not there, ping us!

1.1 Seq66: What!?

Seq66 is a reboot of *Sequencer64*, which is itself a reboot of *Seq24*, a live-looping sequencer with an interface similar to a hardware sequencer. *Seq66* is not a synthesizer. It requires a hardware synthesizer or a software synthesizer. It does not handle audio data, just MIDI.

Seq66 works with *ALSA*, *JACK*, *PortMidi*, and *Windows*. It uses close-to-the-latest C++ features for faster and simpler code.

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 more testing.

1.3 Improvements

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

- Qt 5 as the standard user-interface. Changing the window size works much better.
- A song editor tab for laying out patterns into a complete performance.
- A mutes editor tab, improvements to mutes handling, control, and status display.
- A playlist editor tab, with improved flexibility and automation.
- A sets editor tab.
- A events editor tab for basic fixing of minor event issues.

- A better live frame (main window and external windows) using Qt buttons.
- Non Session Manager support. A sessions tab which shows the current locations of configuration files for the run.
- Repartitioning of the configuration files into separate files for flexibility; added a color palette file, Qt style-sheets, beefed up the 'ctrl' file radically.
- Improved alternate keyboard layout support.
- Mapping of port names to a consistent set of port numbers.
- Palette files and Qt style-sheets can be used to configure the colors of the painted text, lines, patterns, and the size and color of the user-interface items.
- More efficient lookups for controls; lambda functions.

For developers, a *Seq66* build is customizable via C macros, by enabling/disabling options at 'configure' time, and by many command-line arguments. 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. Distro maintainers might create their own build configurations.

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 Let's Go!

Make sure no other sound application is running, for the first run. Start *Seq66* to use JACK for MIDI, or on *Windows*, just run it (`qseq66`, or `qpseq66.exe` on *Windows*); for better trouble-shooting, run it from the command-line at first. The port settings will depend on your system. Provide a MIDI file. 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:

```
$ qseq66 --jack-midi --buss 5 data/midi/b4uacuse-gm-patchless.midi
C:\> qpseq66 --buss 1  data/midi/b4uacuse-gm-patchless.midi
```

This can also be done from a dropdown in the main window.

The "data" directory is an installation directory:

```
/usr/share/seq66-0.90/                    (Linux)
C:/Program Files (x86)/Seq66              (Windows)
```

Some of the files in these directories apply to both operating systems. The configration files are:

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

These are created after the first run of *Seq66*.

If the `--alsa` option is used instead of `--jack-midi`, then the *ALSA* subsystem is used (Linux only). The following figure shows the main window using a light desktop theme and green labels.

Figure 1: Seq66 Main Screen

The *Seq66* main window appears as shown above. This figure has many differences from the *Seq24* main window, but the functionality is similar. *Seq66* behaves better on resizing, and can also be configured to start with its size scaled up or scaled down. Most 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 style-sheets ('qss' files).

We break the discussion into sections for the following groups shown in the figure above:

- **Center Tabs**
- **Application Menu**
- **Main Screen Controls**

The **Live** tab is foremost in the application. It provides a grid of *patterns* (also called *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 of being armed is easy to see from the theme's coloring of activated buttons. In addition, 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 an external window. This tab and all the other tabs will be discussed in more detail, each in its own section. The **Menu** is also described later (see section 2 "Menu" on page 17).

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

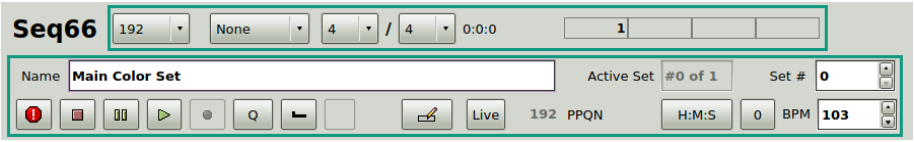


Figure 2: Main Screen Controls

See the following section and section 1.5.2 "Main Bottom Controls, First Row" on page 14.

1.5.1 Main Top Controls (Condensed View)

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:

- **PPQN Selection**
- **Play-set Buss Override**
- **Global Beats Per Measure**
- **Global Beat Width**
- **BBT or HMS Time Display**
- **Beat Indicator**

1.5.1.1 PPQN Selection

This dropdown 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 2.3.1.7 "Menu / Edit / Preferences / Play Options" on page 30.

1.5.1.2 Play-set Buss Override

This dropdown 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 9.4.5 "'usr' File / User MIDI Settings" on page 101), 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 9.3 "'rc' File" on page 89.)

The list of output busses is either the existing MIDI ports on the system, or, if port-mapping (see section 18 "Port Mapping" on page 145) 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. For portability, set the output buss to 0.

1.5.1.3 Global Beats Per Measure

This dropdown 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.

Values: `1 to 16, 32`

1.5.1.4 Global Beat Width

This dropdown 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.

Values: `1 to 16, 32`

1.5.1.5 BBT or HMS Time Display

This text simply shows the current time during playback. It can be shown in BBT (bars:beats:ticks) or HMS (hours:minutes:seconds).

1.5.1.6 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 white. It does not adapt to changes in the time-signature until playback is stopped.

1.5.2 Main Bottom Controls, First Row

The bottom main control items take up two rows. The first row contains:

- **Set Name**
- **Active Set Indicator**
- **Set Changer**

1.5.2.1 Set Name

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

1.5.2.2 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.

1.5.2.3 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.

1.5.3 Main Bottom Controls, Second Row

On to the next section of the main bottom buttons, the second row contains:

- **Panic Button**
- **Stop Button**
- **Pause Button**
- **Play Button**
- **Loop Button** (not shown, new since 0.93.0)
- **Live Record**
- **Keep Queue Button**
- **Mute Group Learn Button**
- **Developer Test Button**
- **Song Editor Button**
- **Song Mode Button**
- **PPQN Indicator**
- **BBT/HMS Toggle Button**
- **Tap BPM Button**
- **Beats Per Minute Control**

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

1.5.3.1 Panic Button

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.

1.5.3.2 Stop Button

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.

1.5.3.3 Pause Button

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.

1.5.3.4 Play Button

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.

1.5.3.5 Loop Button

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.

1.5.3.6 Live Record Button

This button causes a live playing session to be recorded. 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,

1.5.3.7 Keep Queue Button

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.

1.5.3.8 Mute Group Learn Button

Also called the "L" button. Sets up to learn the current set of active patterns ("mute group") into a mute-group. 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*. 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 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 section 12 "Seq66 Mutes Master" on page 120.

1.5.3.9 Developer Test Button

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

1.5.3.10 Song Editor Button

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.

1.5.3.11 Live/Song Mode Button

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 currently no automation control for this button.

1.5.3.12 PPQN Indicator

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

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

1.5.3.13 BBT/HMS Toggle Button

Toggles the format of the current time displayed during playback. It can be shown in B:B:T (bars:beats:ticks) or H:M:S (hours:minutes:seconds).

1.5.3.14 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.

1.5.3.15 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.

2 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 *Non Session Manager*.

2.1 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. The *Seq66* menu entry 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 start under the control of the *Non Session Manager*.

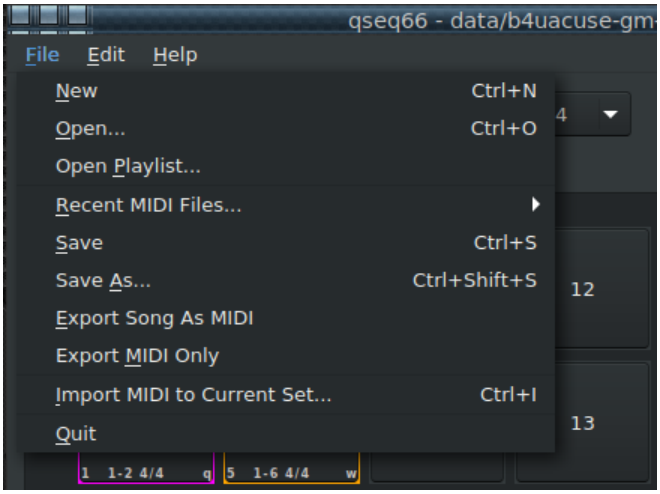


Figure 3: Seq66 Menu File Items

1. **New**
2. **Open**
3. **Open Playlist**
4. **Recent MIDI files**
5. **Save**
6. **Save As**
7. **Export Song as MIDI**
8. **Export MIDI Only**
9. **Import MIDI to Current Set**
10. **Quit** (**Exit** in *Windows*)

For information on the **File** menu when *Seq66* is running under the *Non Session Manager*, see section 7.3.5 "Seq66 Session Management / NSM / File Menu" on page 78.

2.2 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!

2.2.1 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:

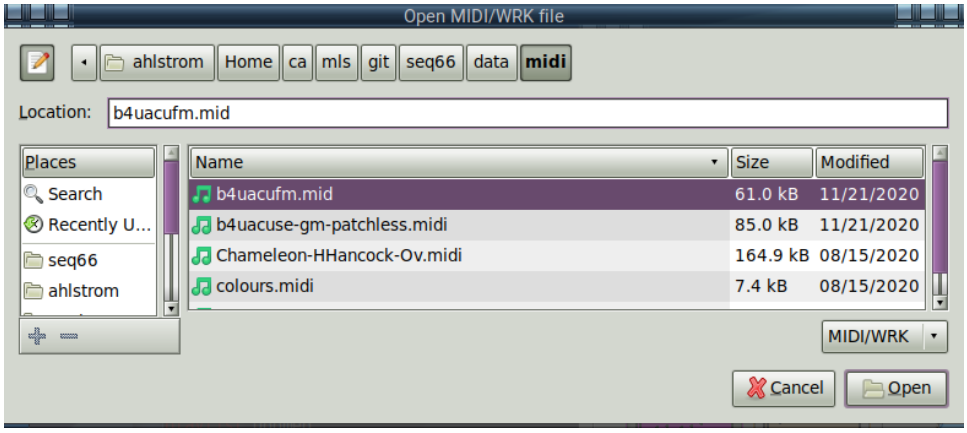


Figure 4: File / Open

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.

2.2.2 Menu / File / Open Playlist

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

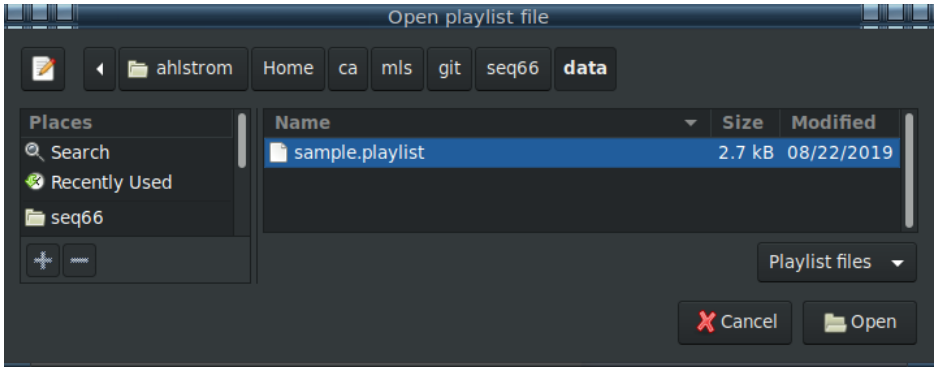


Figure 5: File / Open Playlist

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 10 "Seq66 Play-Lists" on page 113.

2.2.3 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.

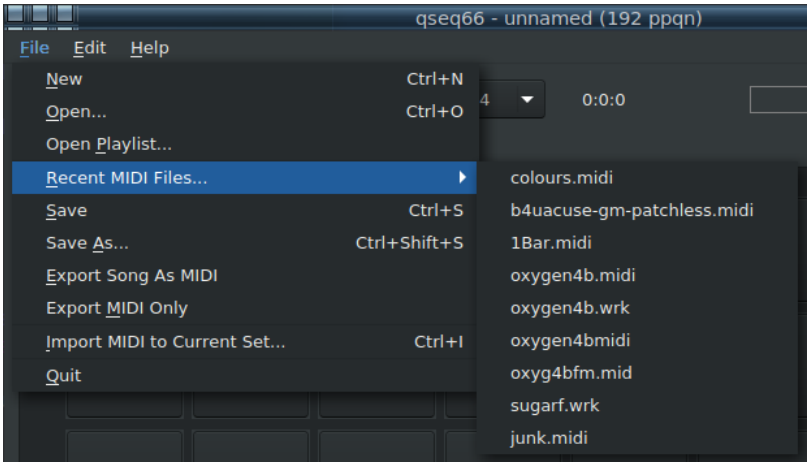


Figure 6: Seq66 Menu File Recent Files

This list is saved in the `[recent-files]` section of the 'rc' configuration file. In the menu, only the last part of the file-name is shown, but 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*. Only unique entries are included in the recent-files list. The limit is 10 recent-file entries. This is a feature from *Kepler34* [6]. One can also set *Seq66* to load the most-recent file at startup. Here is an example from an 'rc' file. Note the startup option.

```
[recent-files]
# Holds a list of the last few recently-loaded MIDI files.
# The first number is the number of items in the list.  The second value
# indicates if to load the most recent file (the top of the list)
# at startup (1 == load it).
3 1
/home/chris/git/seq66/data/b4uacuse-gm-patchless.midi
/home/chris/git/seq66/data/midi/colours.midi
/home/chris/git/Julian-data/TestBeeps.midi
```

2.2.4 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.

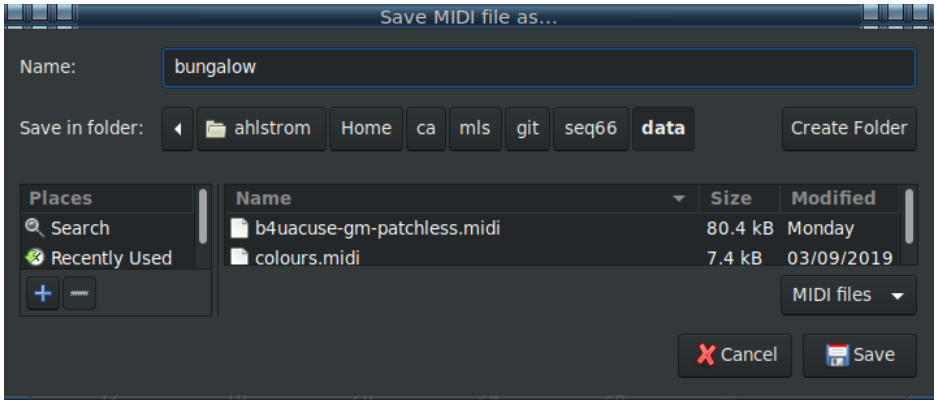


Figure 7: File / Save As

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.)

2.2.5 Menu / File / Import MIDI

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 8.1 "Import MIDI" on page 80.

2.2.6 Menu / File / 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 ?? "??" on page ??.

2.2.7 Menu / File / 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 8.3 "Export MIDI Only" on page 82.

2.3 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** (new with 0.96.1)
10. **Paste To Current Set** (new with 0.96.1)

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** center tab in the main window. The song 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 5 "Song Editor" on page 59. Note that transpose can be enabled in the in the sequence editor (see section 4 "Pattern Editor" on page 41).

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 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. If a set has been marked for the copying of all its patterns to another set, 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.

2.3.1 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.

Missing in this new dialog are: incoming MIDI events to control the sequencer; what keys are mapped to functions; how the mouse works. 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.

2.3.1.1 Menu / Edit / Preferences / MIDI Clock

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.
- 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.

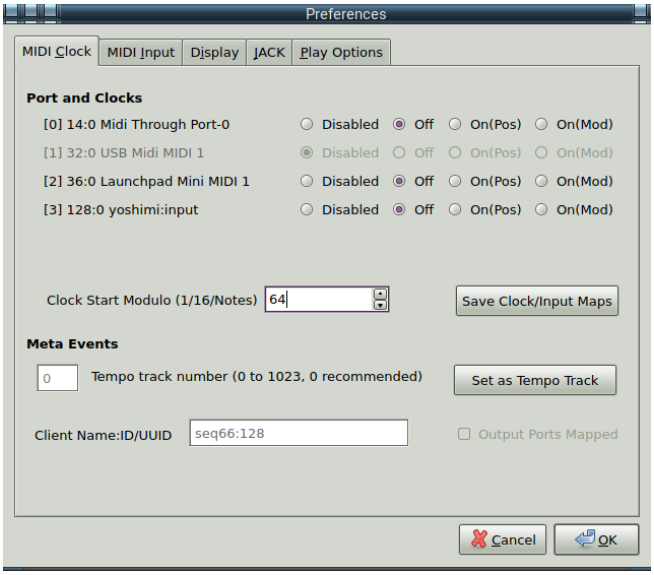


Figure 8: MIDI Clock Tab, ALSA Devices

The following elements are present in this tab:

1. **Ports and Clocks Table**
2. **Clock Start Modulo**
3. **Save Clock/Input Maps**
4. **Meta Events**
5. **Client Name:ID/UUID**

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:

```
[5] 128:4 yoshimi:input
 ^  ^   ^ ^        ^
 |  |   | |        |
 |  |   | |         ---- Port name
 |  |   |  ------------- Client name
 |  |    --------------- Port 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 1.5.1.2 "Play-set Buss Override" on page 13, and section 9.4.5 "'usr' File / User MIDI Settings" on page 101.)

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 16.3 "Seq66 ALSA / Trouble-Shooting" on page 138.

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. Clock Start Modulo (1/16 Notes). This value starts at 1 and ranges up to 16384, and defaults to 64. It is used by the **On (Mod)** setting discussed above. It is the `[midi-clock-mod-ticks]` option in the *Seq66* 'rc' file.

2. Save Clock/Input 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*.

3. Meta Events. This section consists of one item, the Tempo Track number. It 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. *The "Set as Tempo Track" button to the right is not yet functional.*

4. 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.
2. **ID/UUID**. Under ALSA, the client number (client ID) is shown. Under JACK, the UUID that JACK assigned to *Seq66* is shown.

the client ID number assigned to *Seq66* by the ALSA MIDI subsystem. Each instance of Seq66 run under ALSA will have a different client ID.

5. Output Ports Mapped. This read-only check-box shows if the port-mapper is active for the output ports. This item can only be edited by closing the application, editing the 'rc' file, and restarting the application.

There is currently no user-interface item corresponding to the "manual-ports" command-line and 'rc' configuration file option. We should rename this option to "virtual" eventually.

2.3.1.2 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:

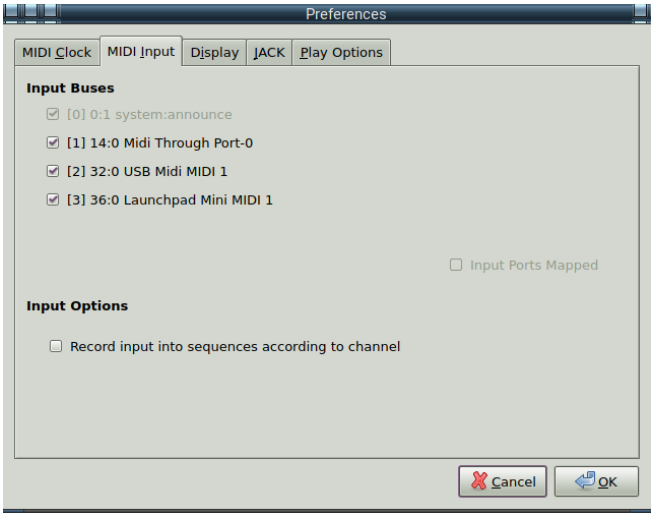


Figure 9: MIDI Input, ALSA View

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.

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. 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.

3. Input Ports Mapped. This stand-alone read-only check-box shows if the port-mapper is active for the input ports. This item can only be modified by closing the application, editing the 'rc' file, and restarting the application.

4. Input Option. **Record input into sequence according to channel** causes MIDI input with multiple channels to be distributed to each sequence according to MIDI channel number. When disabled, the normal recording behavior dumps all data into the current sequence, regardless of channel.

2.3.1.3 Menu / Edit / Preferences / Keyboard

Unlike *Seq24*, *Seq66* does not provide an options tab for setting up the keyboard. 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 20 "Launchpad Mini" on page 151.

For more information on keystrokes, see section 14.1 "Keyboard Control" on page 125.

2.3.1.4 Menu / Edit / Preferences / Mouse

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.

2.3.1.5 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.

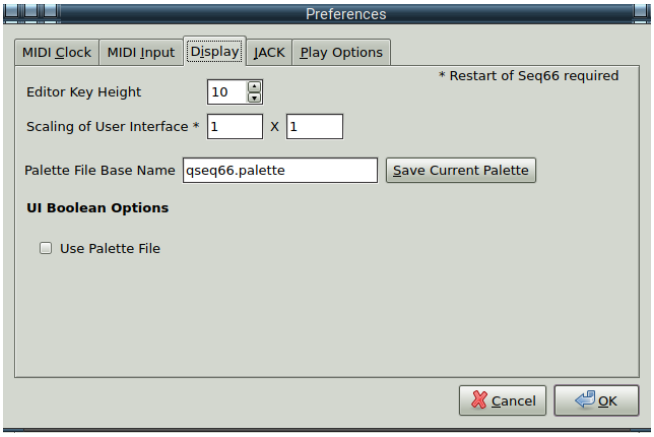


Figure 10: Display Options

Note that this dialog has some new items not shown above.

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. Scaling of User Interface. These two items set scale factor for width and height of the main window. 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.

3. Set Size. 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.

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. If set to 0, the progress box (and its internal pattern color) are not drawn.

5. 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.

6. Palette File Base Name. This text edit holds the base name of a 'palette' file, which is always stored in the *Seq66* configuration directory.

7. 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.

The rest of the options are simple check-box items that are self-explanatory.

2.3.1.6 Menu / Edit / Preferences / JACK

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

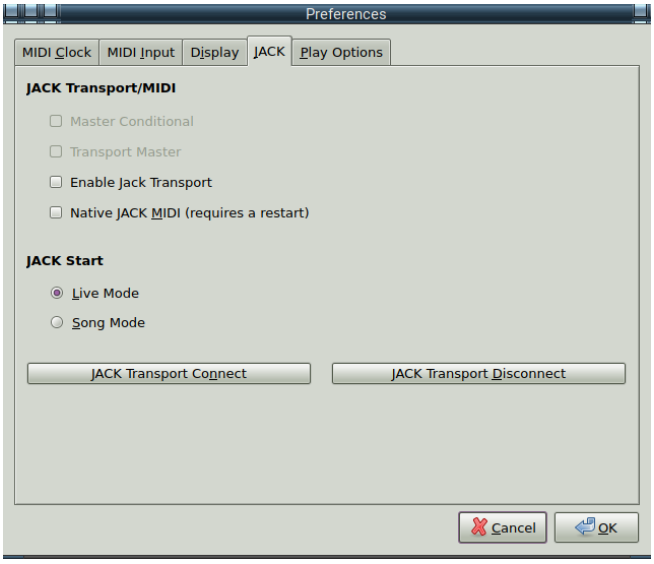


Figure 11: File / Options / JACK

The main sections in this dialog are:

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

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`. **Tip**: *Seq66* generally works better as 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 `seq66` 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. The command-line option is `--jack-midi`.

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.

2.3.1.7 Menu / Edit / Preferences / Play Options

This tab contains some disparate options ostensibly related to playback.

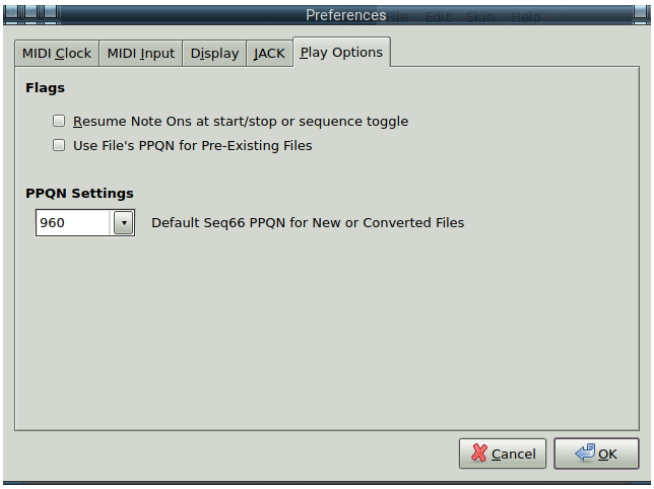


Figure 12: Play Options

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.

2.4 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. It also shows Git version-control information as well.

2.5 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
```

3 Patterns Panel

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 **Patterns Panel** (or **Live Frame** is in the center of the **main window** of *Seq66*. See Figure 1 "Seq66 Main Screen" on page 12. It is here one creates a set of patterns ("screenset"), manages the configuration, controls the playback rate, adds tempo events, and opens the pattern, song, event, mute-groups, or playlist editors.

When the patterns panel has the focus, and *Seq66* is *not* running in **Song** mode, it puts *Seq66* in **Live** mode. 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 part of the song.

If the song editor (see section 5 "Song Editor" on page 59) has the input focus, it automatically controls the muting/unmuting of each pattern, and *Seq66* runs in **Song** mode. (There are ways to override this behavior, such as the **Song**/**Live** button.)

For exposition, we divide the patterns panel into a menu bar, a top panel, a pattern panel (live frame/grid), and a bottom panel. The *Seq66* menu bar is discussed in section 2 "Menu" on page 17.

3.1 Patterns / Main Panel

The main panel of the application provides a grid of empty boxes, as shown in Figure 13 "Patterns Panel Pop-up Menu" on page 33. Each filled box 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 can be 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-driven, labelled **Set** interface item, or by hitting the (default) `Home` key to make it the playing screenset, or by hitting `Page-Up` or `Page-Down` with the pattern window in keyboard focus. 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, depending on configuration.

The `Page Up` and `Page Down`, and `Up/Down Arrow` keystrokes can be used inside of the **Set** spin-button.

It is important to note that, currently, incrementing or 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 11 "Seq66 Set Master" on page 118.

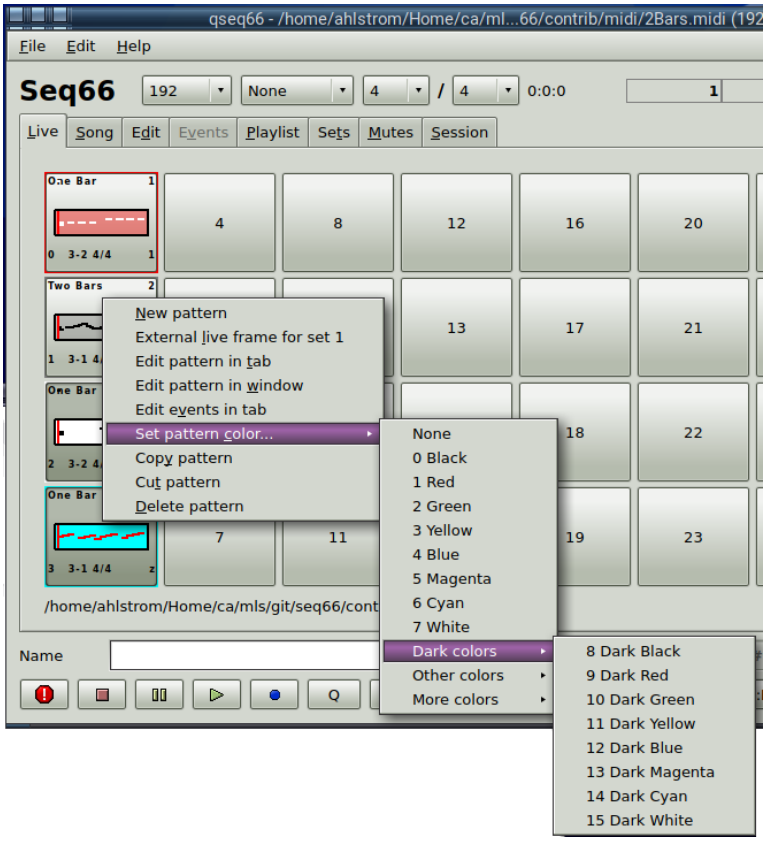


Figure 13: Patterns Panel Pop-up Menu

The individual items annoted in this figure are described in section 3.1.2 "Pattern" on page 35, in more detail.

Observe that feature in the first figure of the next section. The two main items are the empty *pattern slot*, and the slot filled with a MIDI *pattern*:

1. **Pattern Slot**
2. **Pattern**

3.1.1 Pattern Slot

An empty box is a slot for a pattern. If a pattern is present in the slot, text information and notes are drawn on the button. For example, the top line will show the title of the pattern, the number of measures in the pattern, and indicate if the pattern has a loop-count. A right-click over a pattern button brings up a fairly extensive popup menu. Also see section 3.1.2 "Pattern" on page 35.

A pattern can show a number of different statuses based on the coloring of elements in the pattern slot. (However, note that some of the special coloring using in *Sequencer64* is 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. But 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 note useful in reducing the overhead of long patterns.
- **Red events**. Indicates a pattern for which the new 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* [6]. The color is more saturated when the pattern is unmuted. The pattern menu for sequence color is shown in the previous figure. We've used colors in most of the examples in this manual.

Right-clicking on an empty box one brings up a menu to create a new loop, as well as some other operations.

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 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. More on the external live frames later.

Once a new loop is created, there are more options for that slot. A right-click on an already-filled box brings up a menu to allow one to edit it, or perform a few other actions specified in the context menu. Here is that 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. **Set pattern color**
7. **Copy pattern**
8. **Cut pattern**
9. **Delete pattern**
10. **Merge into pattern** (new, not yet shown)

The first two menu entries are 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. Can also be done by double-clicking on an empty slot.

2. External Live Frame for Set. This selection uses the pattern number to open the corresponding screenset number in an external **Live** frame.

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 somewhat simplistic, useful for trouble-shooting a pattern.

4. Edit Pattern In Window. Selecting this item brings up the pattern in an external pattern editor that is more sophisticated than the **Edit** tab.

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 6 "Event Editor" on page 66, 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. Set 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. See section 13 "Palettes for Coloring" on page 123.

7. 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 3.1.1 "Pattern Slot" on page 33. Note that there is no `Ctrl-C` key for this operation in the live (main) window.

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

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

10. 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.

11. Merge Into Pattern. This item is a new feature. 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.

3.1.2 Pattern

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 box with a number of items of information surrounding it. Here are the

items 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 "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. Also, if the loop-count for the pattern is greater than 0, then an **asterisk** is shown. Remember that a pattern loop-count of 0 means the pattern can repeat "forever".
- **Notes or Fingerprint**. Center. 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. 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 31. Note how it varies fastest by row (top to bottom).
- **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**. Bottom right. 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.
- **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.

3.1.3 Pattern Keys and Click

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.

3.1.3.1 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). Below is the default 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. When a snapshot key is pressed, the state of the patterns (armed versus unarmed) is saved. While the snapshot is in force, one can then change the state of the patterns (using the keyboard or MIDI controls, *not* the mouse) to change how the song plays. When the snapshot mode is exited, the original saved state of the patterns is restored.

Unlike in *Seq24*, the `Alt` keys are not used. In addition, the snapshot key acts like a toggle... no need to hold it down. Lastly, there is only one snapshot key slot. Our preference is to use something that does not trigger desktop commands, perhaps "`F11`" or "`F12`", or one of the keys in the keypad. Configured in the 'ctrl' file `[automation-control]` section.

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 is the "queue" functionality. This means that the change in state of the pattern will not take hold immediately, but will kick in when the pattern restarts. This pending state is indicated by coloring the central box of the pattern grey. Please note the "keep queue" functionality and the "one-shot queue" functionality described below.

A light-grey boundary 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. Queue also works for mute/unmute pattern sets ("groups"); in this case, every sequence will toggle its status after its individual loop ends.

We do **not** recommend using `Ctrl` or `Alt` keys for pattern control. They conflict with application or desktop settings. However, if one insists on such hot-key combinations, use the **Menu** button

in the main window to disable the menu. One can also use normal keys to enable queuing. For example, the minus key or the keypad's slash key can be used.

Pressing the "keep queue" hot-key assigned in the 'rc' file activates a "sticky" queue mode. In this mode, pressing a pattern key immediately turns on queuing, instead of mute/unmute. And multiple patterns can be handled in this way at the same time. Keep-queue persists until one clicks the queue hot-key again, or changes the active (viewed) screen-set. "Keep queue" mode is cancelled by pressing the normal queue hot-key. There is also a **Q** button for the same purpose. Also note the "queued replace/solo" functionality, described a bit later.

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. This hot-key can be changed in the **File** / **Options** / **Ext Keys** / **One-shot queue** field.

The "replace" hot-key sets a form of muting/unmuting. When the "replace" hot-key is pressed, then pressing a pattern's hot-key, that sequence is unmuted, and all of the other sequences are muted.

"Replace" is a form of "solo". "Replace" is also implemented via MIDI control, where the MIDI control can be activated, but then the user has to select the desired sequence.

Seq66 provides an extension to the replace/solo functionality that is called "queued-replace" or "queued-solo". In this feature, when the "keep queue" function is activated, 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. Start playback with some patterns on.
2. Press and release the "keep queue" hot-key. This puts the application into "queue" mode. It is indicated via a "**Q**" button.
3. Press and hold the "replace" hot-key.
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 solo.
5. Click the same pattern hot-key again. Observe that the other patterns that were toggled off are now queued to be toggled on at the next loop. Steps 4 and 5 can be repeated endlessly.
6. To end the "queued-replace" mode, click the normal "queue" hot-key. Also, changing the active screen-set ends "queue-replace" mode. It does *not* end normal queue mode, to preserve the behavior found in *Seq24*. One needs to clear the queue mode in order to select another pattern to solo.

Before pressing the "keep queue" key, patterns 33 ("**q**") and 34 ("**a**") are unmuted, while the desired replace pattern, 32 ("**1**") is off. Then the user presses (and holds) the "replace" key, then clicks the "**1**" key. This puts all unmuted patterns, plus the muted replace pattern as well, into queue mode, as shown by the grey panels. When the progress bar reaches the end of the pattern, pattern 32 will go on, and patterns 33 and 34 will go off. If the replace-pattern is already on, it is not queued, as there's no need to turn it on.

If, while in queue mode, the replace key is held and "**1**" is pressed again, the other patterns will be queued, and will turn on again. Thus, the solo status of the replace pattern can be toggled at will, until queue mode is exited by pressing and releasing the normal "queue" key. If the replace key is *not* held down, and another pattern's replace hot-key is pressed, that pattern will be queued normally. If one wants to change the solo functionality to a different pattern, simple hold the replace

key and click on a different pattern. The new arrangement of soloing is memorized. One can clear the queue mode by pressing the normal queue key.

There are more keys defined in the **Keyboard** dialog, and it is worth figuring out what they do, if not documented here. For a couple of short, but good, video tutorials about using arming, queuing, and snapshots, see reference [30].

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 `Shift` key, and left-click the pattern. This will turn off all the other patterns, so that the selected pattern ins the only one playing. Holding the `Shift` key and clicking the same pattern again will unmute all of the other patterns.

3.1.3.2 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 does nothing when the mouse rests inside a pattern box.

3.2 Patterns / Bottom Panel

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. **Song Record**
5. **Song Record Snap**
6. **BPM**
7. **Tap Tempo**
8. **Log Tempo**
9. **Record Tempo**
10. **Keep-Queue Status**
11. **Name**
12. **Set**
13. **Toggle Song Editor**

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. 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.

5. Song Record Snap. This button toggles snapping the beginning and end of a recorded trigger to the nearest beat. There is no hot-key for this button at this time. And, in fact, at present, song-record snap is alway in force.

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. 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.

10. 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.

11. Toggle Song Editor. Pressing this button toggles the presence on-screen of the **Song Editor**. The `Ctrl-E` keystroke can also be used.

3.3 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.

3.4 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.

3.5 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.

4 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.

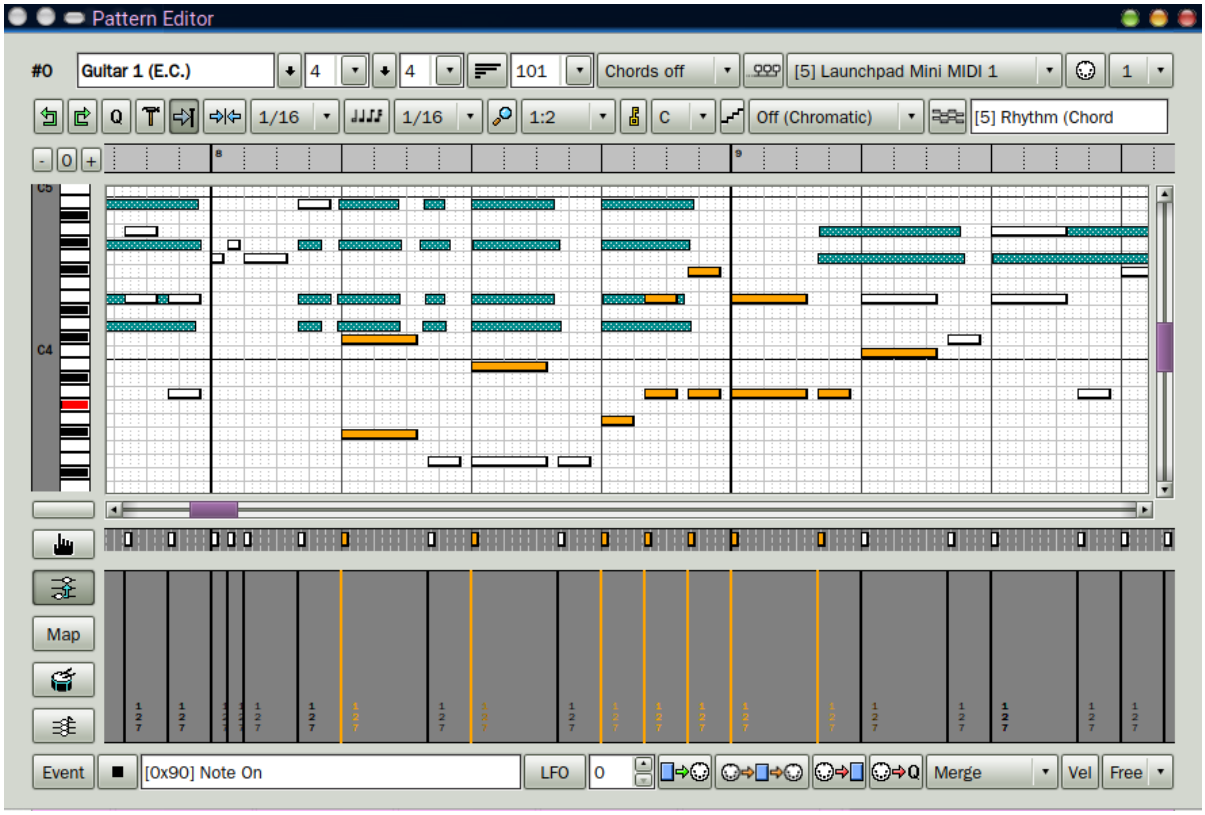


Figure 14: External Pattern Editor Window

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**
6. **Data View**
7. **Bottom Row**
8. **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, 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. Another option is to press the "hand" 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.

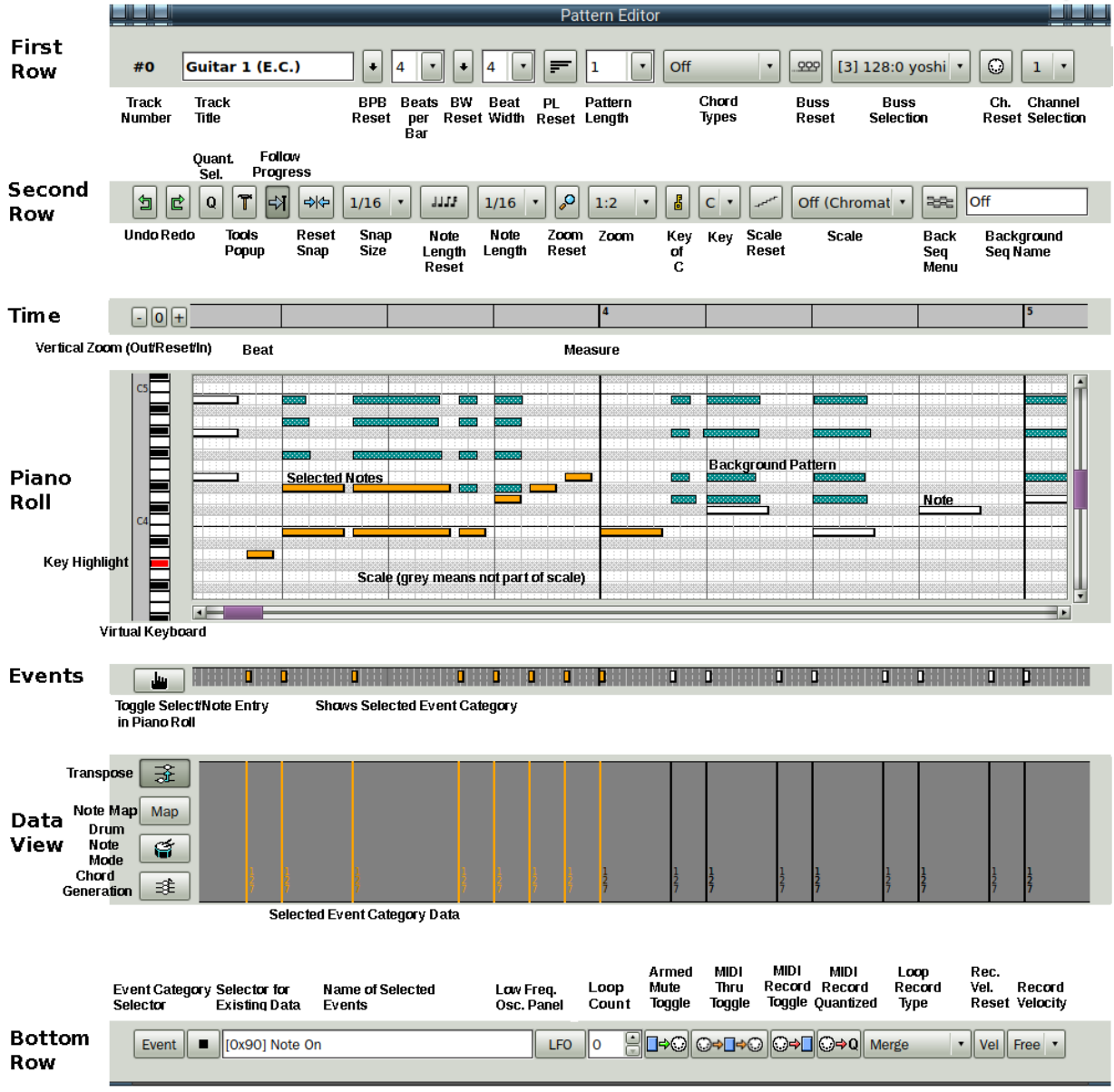


Figure 15: Pattern Editor Window, Annotated

4.1 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 Title**

3. **Beats Per Bar Reset** and **Beats Per Bar**
4. **Beat Width Reset** and **Beat Width**
5. **Pattern Length Reset** and **Pattern Length**
6. **Chord Types**
7. **Buss Reset** and **Buss Selection**
8. **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. 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.

4. 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.

5. 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**.

6. 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 View**.

7. Chord Generation.

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.

4.2 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**. 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.
   - **Inverse selection**, which inverts the selection of notes.
2. **Timing**. This menu offers two ways to tweak the timing of the selected note:
   - **Quantize** quantizes the selected notes, the same way as the **Quantize** ("**Q**") button.
   - **Tighten**, which is merely a less strict form of quantization.
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 Selected**, 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.

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 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 [24].) 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.

4.3 Pattern Editor / Time

The **Time** (or **Measures**) bar provides an explicit count of beats and bars. It also includes vertical-zoom buttons at its left side. As of version 0.93.0, *Seq66* allows the **L**/**R** markers to be placed on

the pattern editor's time-line, to allow for close work with sections deep into a long pattern. This looping now works with both Live and Song modes.

4.4 Pattern Editor / Piano Roll

The piano roll is the center of the pattern/loop/track/sequence editor. It is accompanied by a thin "event bar" ("event area", "event strip") just below it, and a taller "data bar" 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 can show notes, a background pattern, and a scale. Notes are shown as narrow rectanges; the pattern and scale are shown as bars running the whole 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. 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,

If no notes are selected, 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. This can be convenient, especially if the arrow-keys are unwieldly. For example, the *Microsoft Arc* keyboard puts all four arrows on one button!

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 advanced. 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.

4.4.1 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.

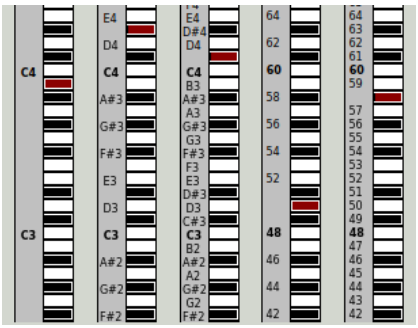


Figure 16: Virtual Keyboard Number and Note Views

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.

4.4.2 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 a little 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 some *Seq66* windows 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. 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.

4.4.3 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**. To deselect the notes, click somewhere else in the piano roll, and the notes should change back to white. There is no way to deselect a single note, with, say, a `Shift-click` or `Ctrl-click` action.

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 4.2 "Pattern Editor / Second Row" on page 45can 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.

4.4.4 Pattern Editor / Other Keys

Here are some other 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 attemtp 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 `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 do not occur unless note-wrap-around occurs. When they do occur, they are painted in magenta. 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.

4.4.5 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.

4.5 Events Editor

Also known as the "events pane" or "events panel". The narrow (a few pixels high) events strip shows discrete events, such as `Note On` and `Note Off`. These and other events appear as small squares in the event strip, along with a black vertical bar in the **Data View** 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 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.

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.

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.

To move the event(s) to a different spot, select it/them via the left button. Then drag the selection as desired. it is currently not possible to move them to positions smaller than the beat size; temporarily reduce the beat size if desired.

The event values can be edited via the data panel, described in the next section.

4.6 Data Panel

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 9.7 "'drums' File" on page 112. 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.

Now on to the **Data View** itself. Also known as the "data pane" or "data panel". **Modify Event Data** offers a way to alter the event data values in the lower pane of the pattern editor, the "data pane". 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.

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. 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 9.4.5 "'usr' File / User MIDI Settings" on page 101.

These values can be easily modified by left-click-dragging the mouse past each line, to chop 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*.

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.

4.7 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. **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, Oneshot)
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.

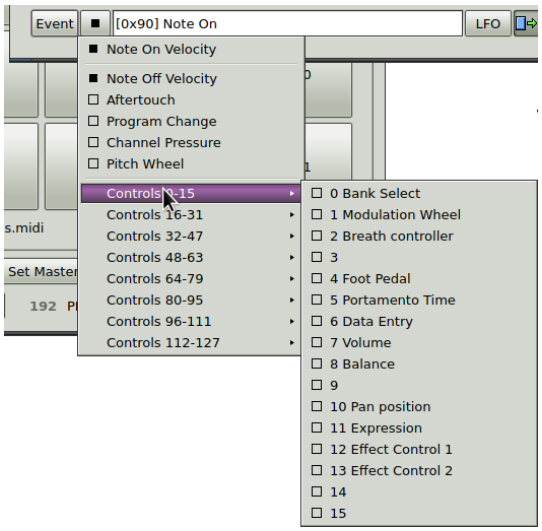


Figure 17: Pattern Editor Event Button Context Menu

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. A low-frequency oscillator allows data events can 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:

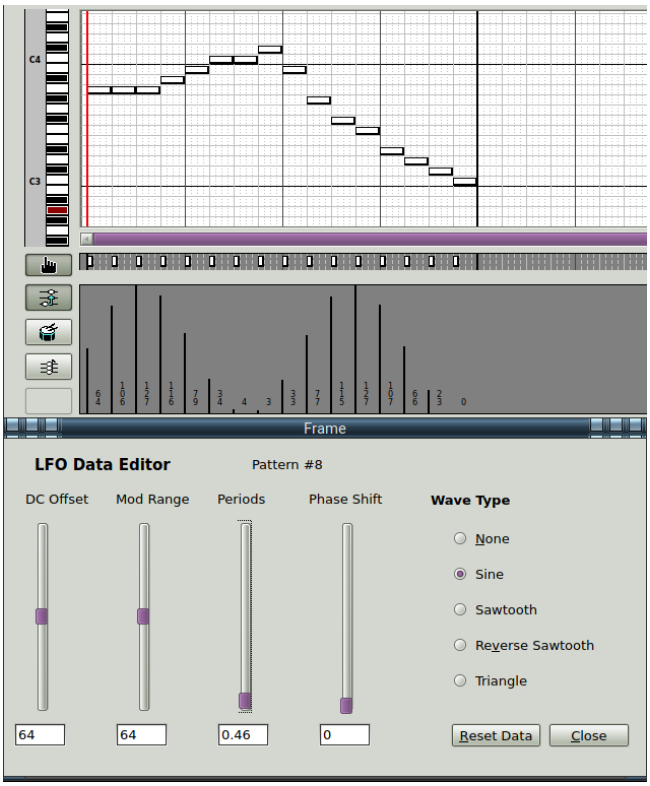


Figure 18: Pattern Editor LFO

1. **DC Offset**. Provides a kind of DC offset for the data value. Starts at 64, and ranges from 1 to 127.
2. **Mod Range**. Controls the depth of modulation. Starts at 64, and ranges from 1 to 127.
3. **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! For short patterns, try a value of 1 at first. For a pattern of one measure in length, this will create one period of the wave.
4. **Phase Shift**. Provides the phase shift within a period of the LFO wave. A value of 1 is a phase shift of 360 degrees (or maybe it is one radian?).
5. **Use Measures**. If checked, a measure is used to calculate the periodicity of a change, rather than using the whole length of the sequence. Useful to modify long patterns.
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.

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.

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.

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 9.4.7.3 "'usr' File / Additional Options / [new-pattern-editor]" on page 104) 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.

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.

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. **Replace.** 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**. 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. This continues indefinitely, whether or not any notes are being played/recorded.
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). At the end of the pattern, recording stops automatically. See the figure below for a recording from a *Yamaha DD-11*. The length of each note is determined by

the snap size for the spacing of drawn events. This feature is still experimental, and bugs are being worked out.

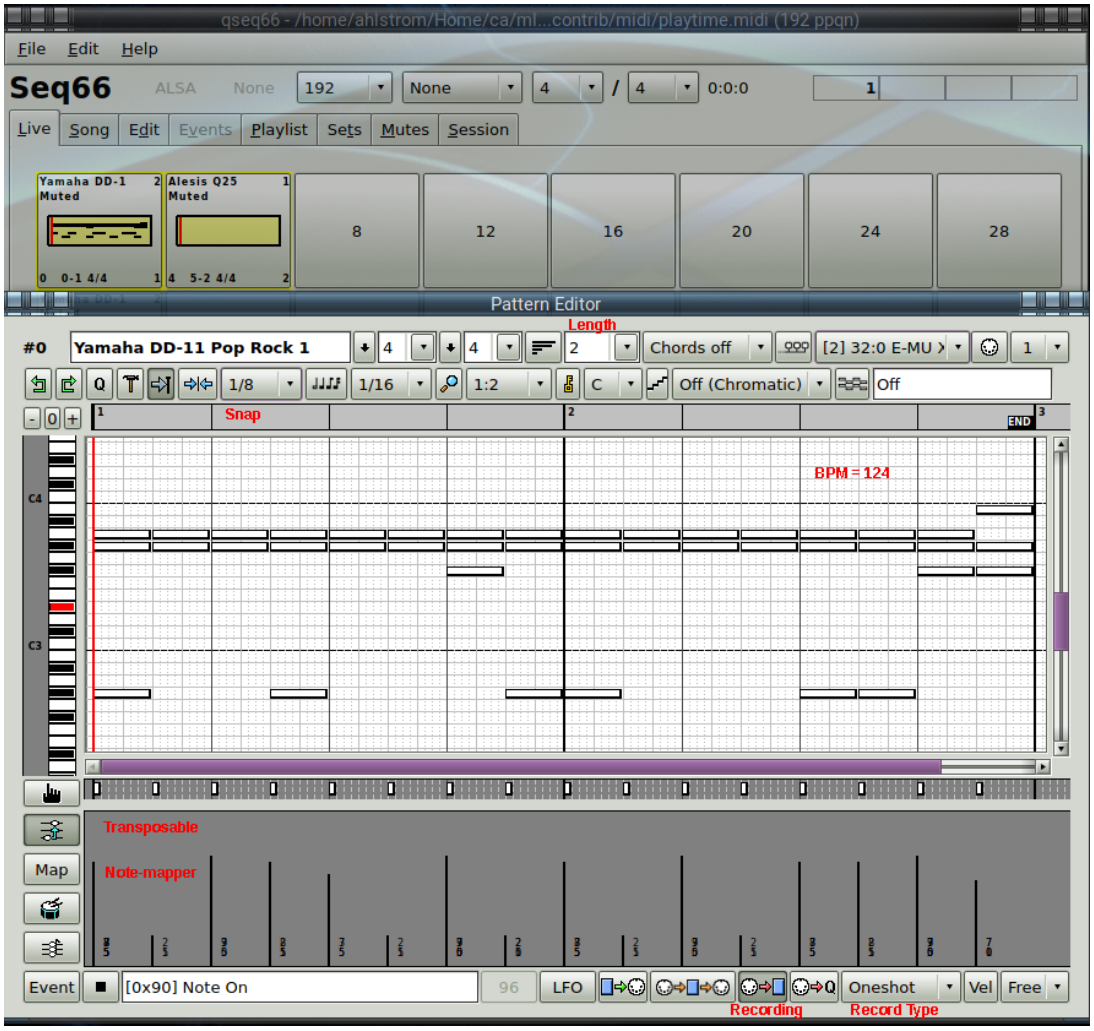


Figure 19: One-Shot Pattern Recording

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 the **Length** to 2 measures (see the red text); the **Snap** to 1/8th; the **Recording Type** to "Oneshot"; and **Recording** on; 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) 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.

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.

4.8 Pattern Editor / Common Actions

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

4.8.1 Pattern Editor / Common Actions / Scrolling

We are still trying to work whether or not to enable this feature in *Seq66*.

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.

4.8.2 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.

5 Song Editor

The *Seq66 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 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 (though with *opposite colors*).

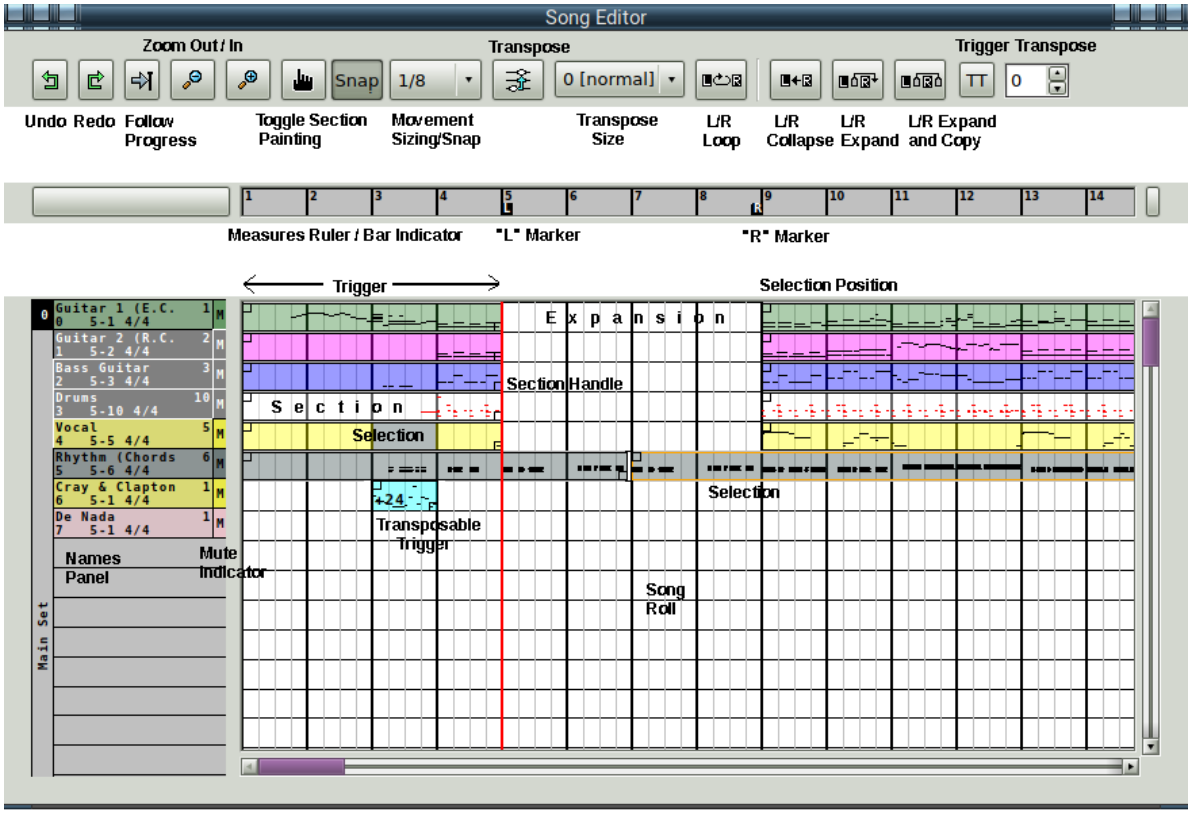


Figure 20: Song Editor Window, Annotated

Note the major items shown:

1. **Top Panel**
2. **Measures Ruler**
3. **Patterns (Names) Panel**
4. **Song Roll**
5. **Bottom Panel**

Not shown in the figure is the estimated duration of the tune; it 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 via buttons and keystrokes

The song editor is not too complex, but for exposition, we break it into the sections enumerated above.

5.1 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. **Trigger Transpose**

1. Stop. Stops the playback of the song. The keystroke for stopping playback is the `Escape` character. It can be configured to be another character (such as `Space`, which would make the space-bar toggle the playback status).

2. Play. Starts the playback of the song, starting 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 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.

Note that there are no stop, pause, and play buttons in this frame. They are supplied by the main window, and the **Song** tab can be activated in the main window.

3. 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`.

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

5. Zoom Out and Zoom In. These buttons change the horizontal zoom. Zoom can also be changed via the keystrokes `z`, `0`, and `Z`.

6. 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.

7. 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`.

8. Record Snap. If enabled, it allows only full clips of a pattern to be added. It turns on record-snap for recording live performance triggers, and it also enables the grid snap functionality noted below. 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.

9. Grid Snap. Indicates the horizontal grid snap for movement actions and trigger drawing. Grid snap determine where the pattern boundaries are drawn. 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:

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

The fractions represent fractions of measures.

The `Length` entry, when **Snap** is on, causes the addition of a trigger to act as in *Seq24*: the pattern is added, but snaps to the length of the pattern so that the new trigger is always an integral number of pattern lengths from the beginning. For long patterns, this feature can result in the trigger beginning well before the click.

Also, in patterns longer than a measure, if `Length` is not selected, the pattern may wrap, so that beginning notes appear at the end of the trigger, and notes can wrap around. When this happens, 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. It is something on our "to-do" list.

10. Transpose. **Transpose** consists of two controls: a drop-down menu to select the transpose direction and amount, and one to apply the transposition. 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.

11. L/R Loop. Activates loop mode. When play is activated, plays the song and loop between the **L marker** and the **R marker**. This button has been moved to the main window. 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, as of version 0.93.0, 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.

12. 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.

13. 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.

14. 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.

15. 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.

5.2 Song Editor / Measures Ruler

The measures ruler ("bar indicator") consists of a *timeline* at the top and the **L marker** and **R marker** mentioned above. There are some hidden details in the measures panel.

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".

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

5.3 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. **Mute Indicator**. The letter M is in a black box if the track/pattern is muted, and a white box if it is unmuted. Left-clicking on the "M" (or the name of the pattern) mutes/unmutes the pattern. 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.
8. **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.

Right-clicking on the pattern name or the "M" button brings up the same pattern editing menu as discussed in section 3.1.2 "Pattern" on page 35. Recall that this context menu has the following entries: **Edit...**, **Event Edit...**, **Cut**, **Copy**, **Song**, **Disable Transpose**, and **MIDI Bus**.

5.4 Song Editor / Song Roll

Also known as the "arrangement panel" or "performance editor". The "Piano 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.

5.4.1 Song Editor / Song Roll / Layout

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. **Trigger Creation**. By click-dragging the mouse 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.
4. **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.
5. **De-selection**. Left-clicking or right-clicking in an empty area of the song roll will deselect the selection.
6. **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.

7. **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.
8. **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.
9. **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` key to enter the paint mode, and `x` escape it. See section 5.4.2 "Song Editor / Song Roll / Keystrokes" on page 65.

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 i(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.

5.4.2 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. However, there are only three levels of vertical zoom: half-size, normal, and double-size. 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. `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.
- 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.
- 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.

5.5 Song Editor / Bottom Panel

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

6 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. 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. It does not present handy dropdown lists for various items.
3. It does not detect any changes made to the sequence in the pattern editor; we will ultimately add a refresh button.
4. It does not have an undo function.
5. 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.
6. 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** 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.

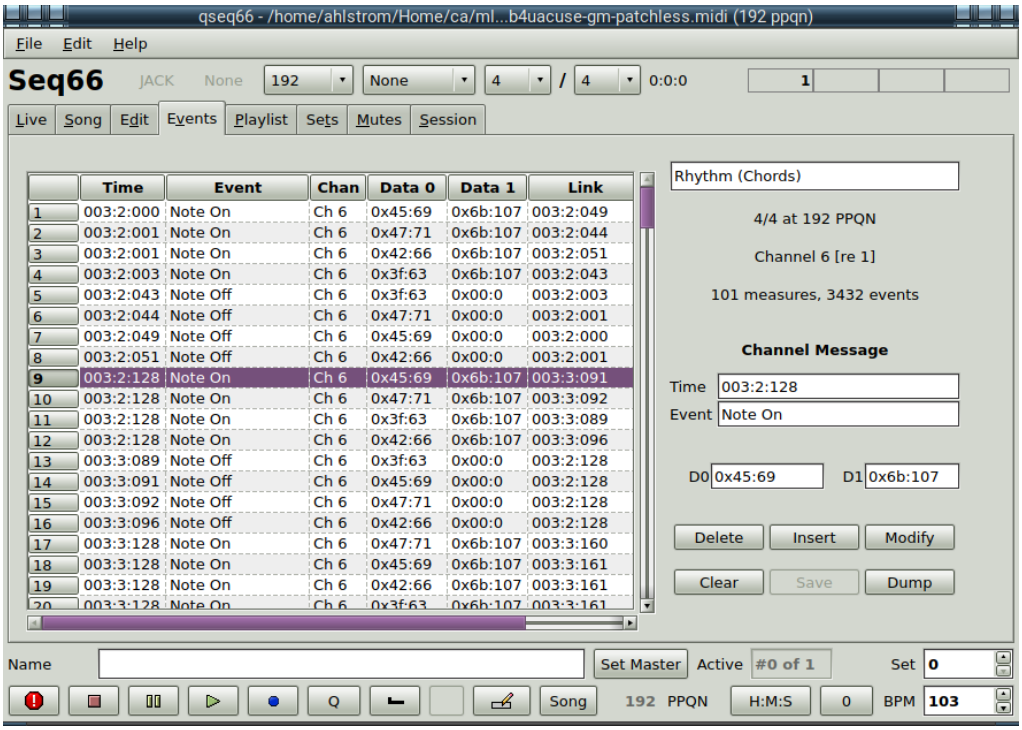


Figure 21: Event Editor Window

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.

6.1 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. 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.

6.1.1 Event Frame / Data Items

The event frame shows a list of numbered events, one per line. The currently-selected event is highlighted in cyan text on a black background. 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 0 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.

3. Event Name. Displays the name of the event. The event name indicates what kind of MIDI event it is. See section 6.3 "Event Editor / Edit Fields" on page 69.

4. Channel Number. Shows the channel number (for channel-events only) re 0, not 1. For the user, of course, 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").

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.

6.1.2 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.

6.2 Event Editor / Info Panel

The "info panel" is simply 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. Five items are shown:

1. **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.
2. **Time Signature**. A pattern property, shown only as a reminder. It can be set in the pattern editor.
3. **PPQN** Shows the "parts per quarter note", or resolution of the whole song. The default PPQN of *Seq66* is 192.
4. **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.
5. **Sequence Count** Displays the current number of events in the pattern. This number changes as events are inserted or deleted.

6.3 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)

Important: changes made in the event editor are *not* written to the sequence until the **Save to Sequence** 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. 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:

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, as of *Seq66* 0.96.1, 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.

4. 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.)

5. 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.

6. Delete. Causes the selected event to be deleted. The frame display is updated to move following events upward.

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.

6.4 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** (events to file)

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`.

Again, good luck with this tab. Bug reports are appreciated.

7 Seq66 Session Management

Session management helps recreate complex setups and provide some uniformity of application control. 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 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. 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. This allows for 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, 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.

If one desires 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 still set up for NSM usage. The *NSM* setting is useful for attaching to a pre-existing known session. Also, *JACK* session management event are only processed if *JACK* is selected. *JACK* session management will still start *Seq66* in an existing session, if *JACK* is not selected, but that's it.

7.1 Seq66 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*, to some extent, 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`.

7.2 Seq66 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:

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

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-midi --jack-master --jack-session-uuid 84670 --home ${SESSION_DIR}
qseq66 --jack-midi --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.

7.3 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 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*.

General, the applications it manages should be installed normally (that is, for system-wide usage, in `/usr/bin/` or `/usr/local/bin`).

7.3.1 Seq66 Session Management / NSM / First Run Without NSM

This section discusses what happen 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 7.3.2 "Seq66 Session Management / NSM / Run in NSM" on page 75.

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.

7.3.2 Seq66 Session Management / NSM / Run in NSM

When *Seq66* is run in *NSM* for the first time, what happens to the new configuration depends on whether or not the normal configuration files exist in the normal (no session) default configuration location.

If *no* configuration files exist, then new configuration files are created in the *NSM* session directory. If these configuration files *do* exist, then they are *replicated* in the *NSM* session directory.

No existing normal configuration. Here, we have just installed *Seq66*, but have not yet run it. We start the *non-session-manager*, create a new session, and add `qseq66` to this session. This starts `qseq66`, and, after a short delay to get the session information from the daemon, a new configuration is created in the session directory.

Existing normal configuration. If there is an existing *Seq66* configuration, then running `qseq66` in a new session will cause the normal configuration to be recreated in the new session directory, including play-lists and MIDI files. If a play-list has been configured, it is also copied, and so are the MIDI files it requires. (Their relative directory structure is preserved.)

If *JACK* has been configured to be used by *Seq66*, be sure *JACK* is started (e.g. by running the *qjackctl* application.)

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 the loopback interfaces are defined. In some versions of *Linux*, it isn't defined properly, and the *NSM* daemon (`nsmd`) will not start. Here is an example for the default install 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.
```

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. 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.mutes
  |   |   +-- qseq66.rc
  |   |   +-- qseq66.usr
  |   +-- midi/
  +-- session.nsm
```

So *NSM* has created a directory with the session name we gave it: `Seq66`. Under that directory is a file, `session.nsm`, which contains information like the following:

```
seq66:qseq66:nXYZT
```

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*.

7.3.3 Seq66 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"
```

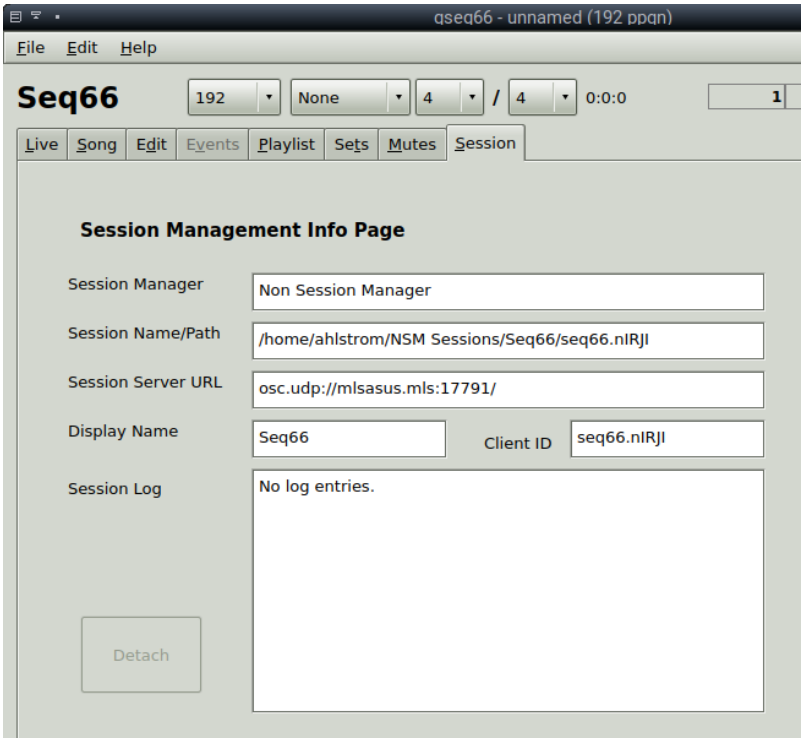
The `url` is not used if running *Seq66* from the *NSM* GUI... the application will get the URL from the *NSM* environment.

Note that `qseq66` can still be run outside of a session manager. It will detect the absense of the session manager and run normally.

7.3.4 Seq66 Session Management / Sessions Tab

The *Session* tab is a *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.

This tab is not yet fully functional and is meant to display information to help the user understand what is happening in the run. In particular, **Detach** and **Session Log** don't yet work.



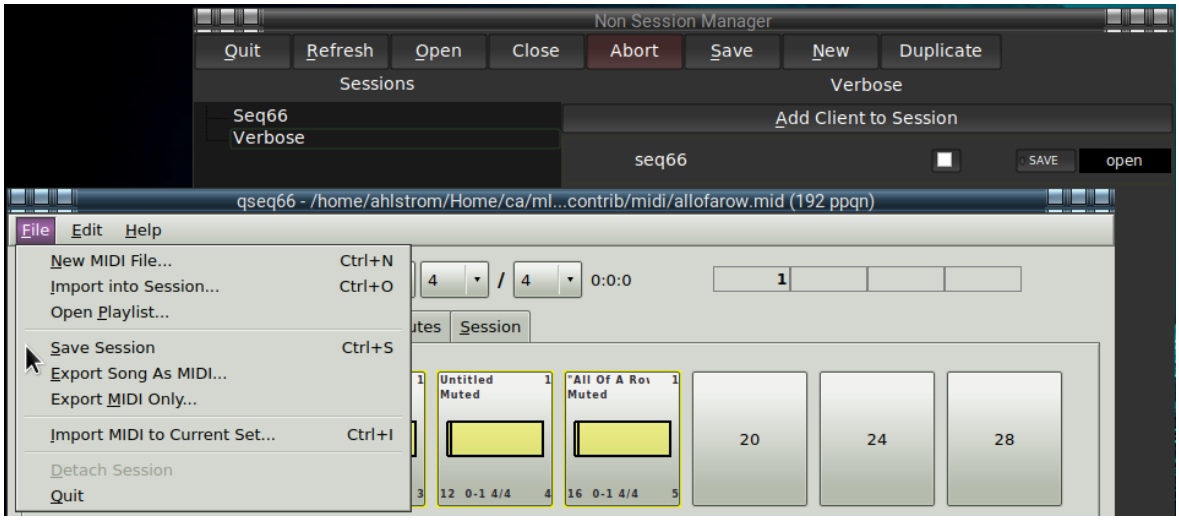
Session Tab When Running Under NSM

This section describes the *Session* tab in the main *Seq66* window. This tab is merely informative and *read-only*. 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. If not running in a session, the active configuration directory (which can be modified via command-line arguments) is shown.
- The OSC URL of the session, 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.
- The log of action of the session manager. Not yet supported, though one can see what is happening by running in a console window.

7.3.5 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 When Running Under NSM

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 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.
- **Open Playlist**. Works the same as without session management, but gets the list from the session directory.
- **Save Session**. This function saves the main configuration files (except for the 'usr' file), saves the play-list and note-mapper files, if in use, and saves the current MIDI file, if any.
- **Export Song As MIDI**. 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 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.
- **Import MIDI to Current Set**. 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.
- **Detach**. Allows *Seq66* to detach from session management. This process simply disconnects from *NSM* and restores the normal *File* menu. Currently, it is disabled because it causes issues that we cannot yet solve.
- **Quit**. Quits *Seq66*. There are no messages to send to *NSM* in this case. The `nsmd` daemon detects that the *Seq66* client has disappeared (and notes that on the console output, if available).

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 7.1 "Seq66 Session Management / Signals" on page 72.

7.3.6 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.

7.4 Seq66 Session Management / LASH

LASH support has been removed. Use the *NSM Session Manager* or the *JACK Session Manager*.

8 Import/Export

This section explains the details of the MIDI import and export functionality, accessed by the main menu as noted in sections 2.2.5, 2.2.6, and 2.2.7, on page 21.

8.1 Import MIDI

The **Import** 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'

8.2 Export Song as MIDI

Thanks to *Seq32*, exporting song performances (see the **Song Editor**) to standard MIDI format has been added. The **Export Song as MIDI** 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 present in the **Song Editor**, and, *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:

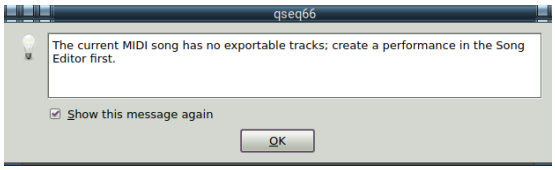


Figure 22: MIDI File Unexportable

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.

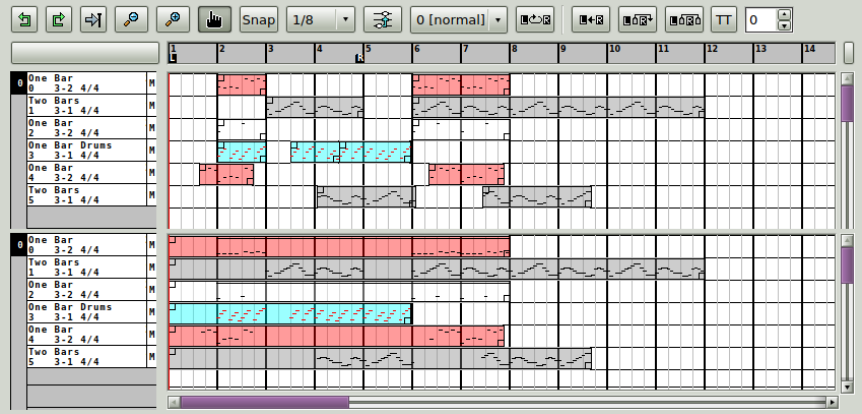


Figure 23: MIDI File Layout Before/After Export

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.

8.3 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 **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.

8.4 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 8.2 "Export Song as MIDI" on page 81), the tracks to be exported (combined into a single track) must be unmuted and have layouts in the song editor.

This action removes all the existing tracks and merges them into track 0.

9 Seq66 Configuration

Seq66 configuration has become more elaborate with time, with more configuration files (*feature-itis?*). Fortunately, 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.
- **'palette' File**. This new file allows replacing the default piano-roll, time, data, and events drawing to be tailored.
- **'playlist' File**. This new file specifies a file that contains one or more playlists and MIDI controls for them.
- **'qss' File**. Qt style-sheets can now tailor the appearance of theme-drawn elements (e.g. the pattern-grid buttons and text controls).

After the first run of *Seq66*, it will generate a set of configration files in the default *configuration* directory, with names dependent on the version of *Seq66* being run:

```
/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
```

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 'usr' file's "user UI tweaks" 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* (see section 7 "Seq66 Session Management" on page 71), 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 13 "Palettes for Coloring" on page 123.

These files contain the the data for remote MIDI control, computer keyboard control, MIDI clock, JACK transport, and a many other settings.

Seq66 always overwrites the most of these files upon exiting. One must therefore quit *Seq66* before making manual modifications to these files.

Some of the settings can be modified in the **Edit / Preferences** dialog, or overridden from the command line.

9.1 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.

9.1.1 [Seq66] Section

This section is generally just 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`.

9.1.2 [comments] Section

This 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.

9.1.3 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.

9.1.4 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 = value` style described below, but there are a few places where the old style persists.

9.1.5 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.

Also note that there are still a lot of old-style variable specifications in place. As time allows, they will be fixed, one-by-one, in a manner that does not break old configurations.

9.1.6 Stanzas

Please note that, as of version 0.92.0 of *Seq66*, we have streamlined the control-file stanza by eliminating the "enabled" and "channel" columns in a stanza, 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 bracketed by square brackets, and some optional data at the end. The values inside the square brackets are numeric, and can be in decimal format, sometimes hexadecimal format, or in binary "0/1" format.

```
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
. . .
```

See section 9.5.2 "'ctrl' File / Loop Control" on page 105, which describes the details of this layout.

9.2 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 the built-in NSM (*Non Session Manager*) support.

`-T --no-nsm` Ignore the NSM setting in the 'usr' file. ('T' for 'typical').

`-X --playlist [filename]` This option loads the given file-name as a play-list file. See section 10 "Seq66 Play-Lists" on page 113.

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.

`-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`

`-H --home [directory]` Change the "home" configuration directory from `$HOME.config/seq66` The configuration files are loaded from or saved to the specified directory.

`-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 9.3.10 "'rc' File / JACK Transport" on page 92.

`-t --jack-midi` *Seq66* will run with JACK if JACK is running.

qseq66.rc: `[jack-transport] jack-midi = true`

`-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 7.2 "Seq66 Session Management / JACK Session" on page 72.

`-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`

`-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.

`-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`.

`-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`. Makes the `seq66cli` application fork to the background, or makes the `seq66cli` application run in the foreground so that console output can be seen.
- `log=filename.log`. Reroutes standard error and standard output messages to the given log-file. This file is located in the configuration directory. If this file is present, additional log information is appended. The default log-file name is specified in the `[user-options]` section of the 'usr' file.
- `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. 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.

Many of the above options are including in the relevant configuration files. We will point to them in the future.

9.3 '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:

```
verbose = false
sets-mode = normal
port-namng = short
```

`verbose` is the same as the `--verbose` command-line option, except that, if set to "true", it affects every invocation of *Seq66*.

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 values are 'short' or 'long'. The short style just shows the port number and short port name; the long style shows all the numbers and the long port name. For *Windows* (portmidi), the 'long' option works better.

9.3.1 '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 is 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 9.5 "'ctrl' File" on page 104.

9.3.2 '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 sections 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.

9.3.3 'rc' File / Color Palette

```
[palette-file]
active = false
name = "qseq66.palette"
```

The only need for a palette file is when the user is 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.

9.3.4 '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 19 "One-Shot Pattern Recording" on page 57, and the surrounding discussion.

9.3.5 '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 18 "Port Mapping" on page 145; it contains more details.

9.3.6 'rc' File / MIDI-Clock Section

The MIDI Clock fields contain the clocking state from the last time *Seq66* was run, and their status, and their names. Turn off the clock with a 0, or 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 section has entries for each MIDI output buss.

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"
. . .
```

Note that the 'rc' file has a port-naming option, as described earlier.

9.3.7 '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-channel = false
```

The record-flag is kind of an outcast. Not sure why it is here, as opposed to the 'usr' file; something to rectify later.

9.3.8 'rc' File / MIDI-Meta-Events Section

The new MIDI Meta events section is the start of additional options supporting meta events as normal events in *Seq66*. This section defines just one feature of MIDI meta-event handling 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. If 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**. However, to accommodate existing tunes and their set arrangement, we allow the user to and change the tempo track to another pattern.

9.3.9 'rc' File / Keyboard Control Section

The keyboard control section has been merged into the MIDI control section and been moved into the 'ctrl' file. There is no longer any 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.

9.3.10 '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 9.2 "Command Line" on page 86.

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'
```

`transport-type` is independent of the MIDI playback/record engine (ALSA versus JACK). The '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, and each uses a different JACK client internally.

`song-start-mode` is set to 'true' or 'song' to force the usage of the track layout in the song editor (see section 5 "Song Editor" on page 59). 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'.

9.3.11 '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 would equal 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"
```

Again, see the 'rc' file itself for more information.

9.3.12 'rc' File / Manual ALSA Ports

The name of this setting is a bit of a misnomer in a couple of ways:

1. It actually refers to the usage of *virtual* MIDI ports. These are ports that are 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. 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.

9.3.13 'rc' File / Reveal ALSA 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.

9.3.14 '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.

9.3.15 '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.

9.3.16 '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/"
```

9.3.17 '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"
```

9.3.18 '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 10 "Seq66 Play-Lists" on page 113. It describes the setup, layout, and usage of a *Seq66* playlist file containing one or more playlists.

9.4 'usr' File

This section describes the *Seq66* 'usr' (or "user") file. 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, and allows for some tweaking of the application. This configuration overrides the default values of the **Event** drop-down list and menu items in the Pattern editor, and make 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.

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 the *Seq66* application, and if they are 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:

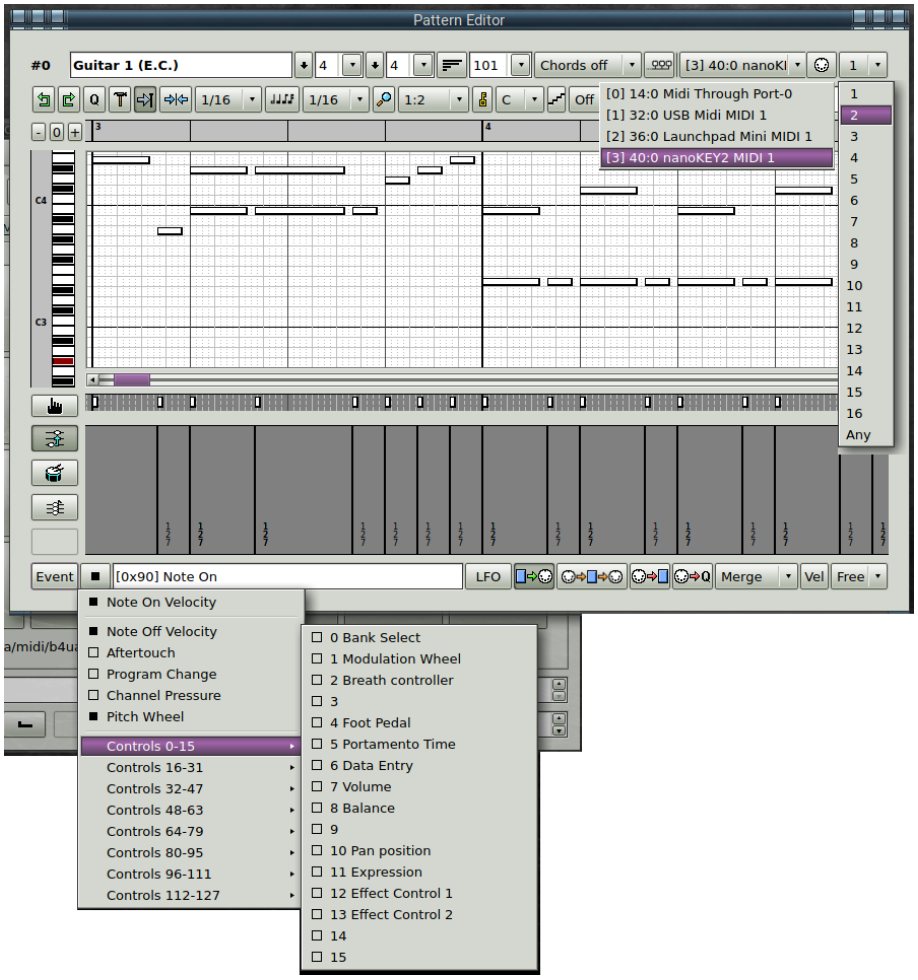


Figure 24: Seq66 Composite View of Native Devices

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 any controllers that the selected MIDI buss 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 defaults, we can create a 'usr' file to set them up. 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 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:

1. Model 2x2 A
2. Model 2x2 B
3. PCR-30

then assume that we have nine different MIDI instruments in our kit. let's number them, too:

1. Waldorf Micro Q
2. SuperNova
3. DrumStation
4. TX81Z
5. WaveStation
6. ESI-2000
7. ES-1
8. ER-1
9. 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 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 1 to 8
   - TX81Z: channels 9 to 11
   - Waldorf Micro Q: channels 12 to 15
   - DrumStation: channel 16
2. Bus 1: Model 2x2 B
   - WaveStation: channels 1 to 4
   - ESI-2000: channels 5 to 14
   - ES-1: channel 15
   - ER-1: channel 16
3. Bus 2: PCR-30
   - TB-303: channel 1

We use the **'usr' configuration file**. to show these items with the proper names associated with each device, channel, and controller value

The *Seq24* configuration file was called `.seq24usr`, and it was stored in the user's `$HOME` directory. *Seq66* uses a new file-name to take its place. 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).

The 'usr' file allows one to give an alias to each MIDI bus, MIDI channel, and MIDI control codes, per channel. 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.

The first important section in the 'usr' file is `[user-midi-bus-definitions]`. 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 ALSA is the MIDI engine being used).

Each of the busses contains 16 (0 to 15) channel entries. These channels are referred to as "instrument numbers", and are represented as and linked to "instruments" in this section:

```
[user-instrument-definitions]
```

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.

So, each instrument is setup as a "channel" in a particular "buss". In the Pattern Editor, when a particular buss and channel is selected, the **Event** menu entries should match the controller entries set up in the 'usr' file.

The list of devices and channels shown earlier can be seen in the *Seq66* sample file `sample.usr`. Deducting 1 from each device number and channel number (so that numbering starts from 0), and consulting the device manuals to determine the controller values supported, one can assemble a 'usr' configuration file that makes the setup visible in *Seq66*.

Peruse the next couple of sections to understand a bit about the format of this file, following along in the sample 'usr' file.

9.4.1 '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.

```
[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 section:

```
[user-midi-bus-0]
2x2 A (SuperNova,Q,TX81Z,DrumStation)
16
0 1
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
```

These instrument-definition sections 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)

9.4.2 '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 instrument
```

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.

```
[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:

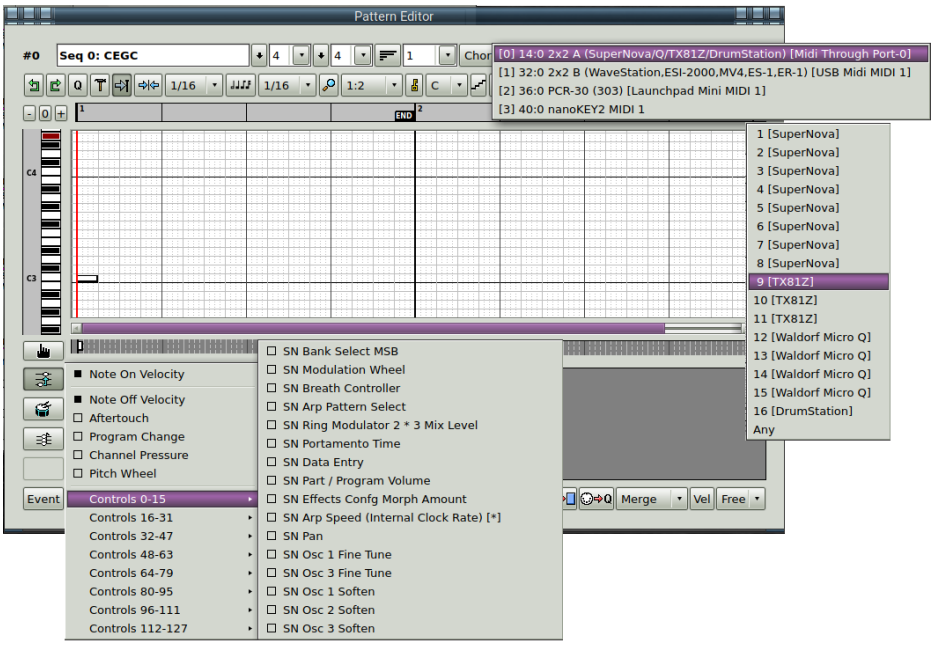


Figure 25: Seq66 Composite View of Devices As Set in "sample.usr"

9.4.3 '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]
mainwnd-rows = 4
mainwnd-columns = 8
mainwnd-spacing = 2
default-zoom = 2
global-seq-feature = true
progress-bar-thick = true
inverse-colors = false
window-redraw-rate = 40
window-scale = 1
window-scale-y = 1
```

There are a number of additional user-interface options. See the generated or sample 'usr' file for descriptions. Also see the chapter on palettes.

9.4.4 '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. And *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.

```
[user-midi-ppqn]
default-ppqn = 192
use-file-ppqn = false
```

It is probably best to set `use-file-ppqn = true`, but that is up to the user.

9.4.5 '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 1.5.1.2 "Play-set Buss Override" on page 13), 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.

9.4.6 '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"
```

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. However, currently we have issue with this option, so one should run `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.

9.4.7 '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.

9.4.7.1 '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
      style-sheet = "qseq66.qss"    # optional, can include a path
      fingerprint-size = 128
      progress-box-width = 0.8
      progress-box-height = 0.4
      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 now uses the new pattern editor in the 'Edit' tab. When used in the **Edit** tab instead of an external window, it is shrunk slight vertically, but the bottom row of buttons is invisible unless the user makes the window taller. The old smaller pattern editor is less functional and a bit more buggy. It has gone away completely.

The following options adopt the new convention for setting variables, in which the format is `item-name = value`. This style will eventually be used for all settings.

The `note-resume` option, if active, causes any notes in progress to be resumed when the pattern is toggled back on.

The `style-sheet` option, if non-empty, causes a user-designed style-sheet to be applied. This is useful in expanding the tab-sizes, for example. The style sheet file is assumed to reside in the normal *Seq66* configuration directory. However, it can have a path component so that the same style sheet can be used by many applications more readily. Here is an example using the style sheet file `data/samples/qseq66.qss`:

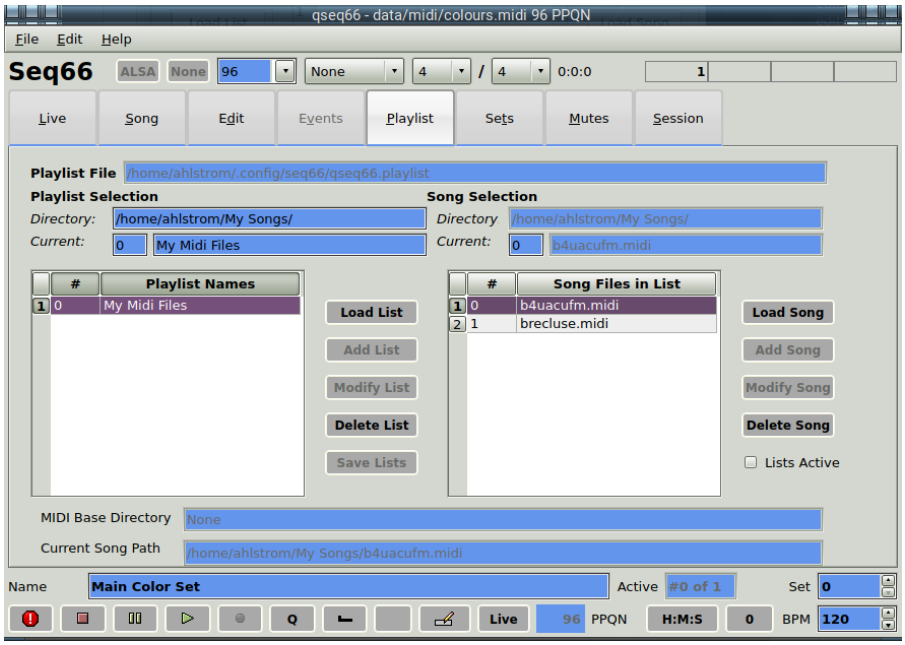


Figure 26: Seq66 View with Style Sheet Applied

Note the blue text fields. Also note the larger tabs, which could be useful on a touch-screen.

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 values

range from 0.0 to 1.0 (full size). If either is 0, then the box isn't 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.

9.4.7.2 'usr' File / Additional Options / [user-session]

`[user-session]` provides a way to cooperate with the *Non Session Manager*.

```
[user-session]
session = none
url = ""
```

See section 7.3.3 "Seq66 Session Management / NSM / Run with Remote NSM" on page 77; it goes into great detail.

9.4.7.3 '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]
armed = false
thru = false
record = false
qrecord = false
record-style = merge
wrap-around = false
```

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 after the pattern loops back to the beginning.

If false, then two unlinked note events appear in the piano roll, colored magenta. The "u" command in the piano roll will remove them.

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.

9.5 'ctrl' File

Like *Seq24*, *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` [12]. Also see the tutorial section section 20 "Launchpad Mini" on page 151. 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 is specfied in the 'rc' file as noted earlier.

9.5.1 '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]
load-key-controls = true
load-midi-controls = true
control-buss = 3              # or 0xff
midi-enabled = true
button-offset = 0
button-rows = 4
button-columns = 8
keyboard-layout = qwerty
```

- `load-key-controls`. Generally, this should always be set to "true".
- `load-midi-controls`. This one could plausibly disabled, but for live performance will certainly be set to "true".
- `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.
- `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 "qwerty" and "azerty". 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.

9.5.2 '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 number is an index number, starting at 0. It indicates what loop the control line will affect. 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 [13].

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 <
```

9.5.3 '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.

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 <
```

9.5.4 '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.

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
```

Note that "Quit" is not a

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".

9.5.4.1 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.

9.5.4.2 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.

9.5.4.3 Automation / Mod Replace

This control the "replace" flag. Then, 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.

9.5.4.4 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. Thus, two MIDI events need to be allocated to this functionality.

9.5.4.5 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 [12], 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 include in the data directory are sample 'ctrl' files for other devices.

9.5.4.6 Automation / Mute Group

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 12 "Seq66 Mutes Master" on page 120.

9.5.4.7 Automation / Screen-Set Play

This MIDI control sets the playing screen-set.

9.5.5 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.

9.5.6 '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.

```
[midi-control-out-settings]
set-size = 32
output-buss = 1
midi-enabled = true
button-offset = 0
button-rows = 8
button-columns = 8
```

The first section sets up some general settings.

- `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.
- `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 20 "Launchpad Mini" on page 151; it showns a fairly comprehensive setup based on the file `data/linux/qseq66-lp-mini.ctrl`.

9.5.7 '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:

```
[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
```

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 the keyboard.

QWERTZ. We don't currently supply this layout, 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"
```

After 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 key that follows, and will not emit a useable key code.

Lastly, 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.

9.6 'mutes' File

This file starts with:

```
[mute-group-flags]
load-mute-groups = midi       # load the mute-groups from MIDI file
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 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.

9.7 '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.

9.8 'palette' File

This file is described in the chapter on palettes, section 13 "Palettes for Coloring" on page 123.

9.9 'playlist' File

This file is described in the chapter on playlists, section 10 "Seq66 Play-Lists" on page 113.

10 Seq66 Play-Lists

Seq66 supports play-lists. A play-list is a variation on the 'rc' file, conventionally ending with the extension `.playlist`. It contains a number of "playlist" sections, each with a human-readable title, 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 songs, 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. it describes the general usage of the [midi-control] section. Using MIDI control makes it possible to use the `seq66cli` headless version of *Seq66* in a live setting. 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. If it is specified on the command line, that playlist setup will be written to the 'rc' file. It can be removed by specifying a blank (i.e. two double-quotes, "") play-list name. The file extension is `.playlist`.

The Qt user-interface supports editing of the play-list, though not yet perfected; bugs still exist. 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.

10.1 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 that the user can edit with a text editor. It can also have an optional comments section, much like the 'rc' and 'usr' files. It is *not* overwritten when *Seq66* exits.

```
[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-new-song = true   # a new song selection unmutes its patterns
deep-verify = false      # If true, every MIDI song is opened and verified
```

The first option allows the load of the next song to enable the patterns in that song. The second 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 each MIDI file exists.

Following the options section are one or more `[playlist]` sections. Here is the layout of a sample playlist section.

```
[playlist]

# Playlist number, arbitrary but 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. This number can be any non-negative value. However, in order to use MIDI control to select the playlist, this number should be 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. 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. Also, 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, it is recommended to keep them between 0 and 127, for usage with MIDI control. And the songs are ordered by this number, rather than by their position in the list.

10.2 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]
# Provides a configured play-list and a flag to activate it.
0     # playlist_active, 1 = active, 0 = do not use it
# Provides the name of a play-list.  If there is none, use '""'.
# Or set the flag above to 0.
"/home/ahlstrom/.config/seq66/sample.playlist"
```

This setup allows a play-list file to be specified and activated. If the name of the play-list file does *not* contain a directory, then the play-list file is search 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.

10.3 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:

```
           Pick-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
```

Both lines specify setting the next playlist or song according to a number, 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, 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.

10.4 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 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 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.

10.5 Seq66 Play-Lists / Verification

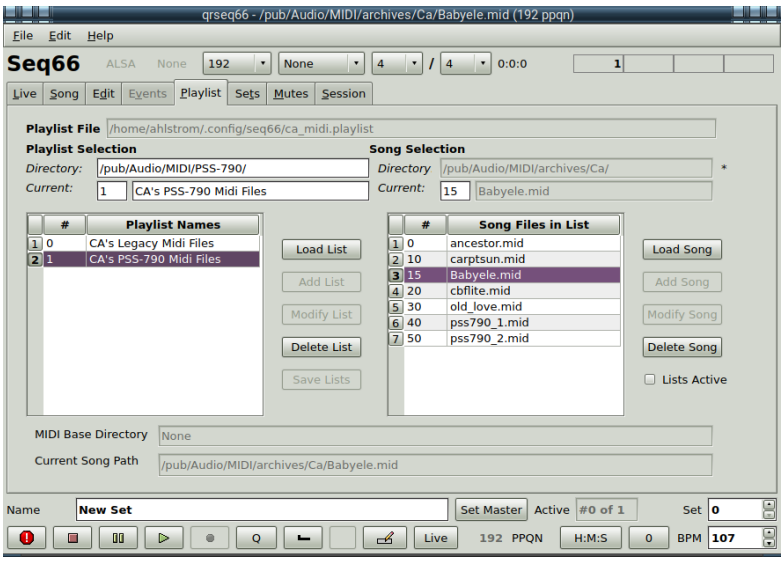
When *Seq66* loads a play-list file, an option allows every song in the play-list file to be verified by loading it. If any load fails, then the playlist will fail to load. This check can be slow when there are many large MIDI files specified in the play-list file.

10.6 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.



Qt 5 Playlist Tab

There is a lot to talk about in this tab.

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 Selection**. These fields display the main MIDI-file directory, the MIDI control number, and the name of the selected play-list. The **Directory** is where the MIDI files reside by default. A file-name can include a different path, however. These fields are editable, with the intent to use them to add a new play-list or modify the current one.
3. **Song Selection**. These fields display the MIDI-file directory, the MIDI control number, and the file-name of the selected play-list. 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. Only the MIDI control number field is editable, with the intent to use it to add a new song.
4. **List Names**. This table shows the MIDI-control number and the name of each play-list.
5. **List Buttons**. These buttons are described below. Please note that, in some cases, the exact functionality is still being worked out or perfected.
6. **Song Files in List**. This table shows the MIDI-control number and the name of each song.
7. **Song Files in List Buttons**. These buttons are described below. Please note that, in some cases, the exact functionality is still being worked out or perfected.

10.6.1 Seq66 Play-Lists / User Interfaces / Playlist Buttons

This section briefly describes the "List" buttons to the right of the play-list table.

- **Load List**.
- **Add List**.
- **Modify List**.
- **Delete List**.
- **Save Lists**.

1. Load List. This button brings up the "Open play-list file" dialog, using the *Seq66* configuration directory as the default directory. It is recommended to use only this directory, especially when running in a session manager. If loaded from somewhere else, save the file back to the configuration directory.

2. Add List. This button is enabled with the editing of the **Playlist Selection** fields. Once these three fields are correct, the list can be added. The new list can then be populated with songs.

3. Modify List. This button is enabled with the editing of the **Playlist Selection** fields. Once these three fields are correct, the list can be modified. **Bug: Currently does not work.**

4. Delete List. 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 a play-list file.

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.

10.6.2 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.

11 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. It is still a work in progress.

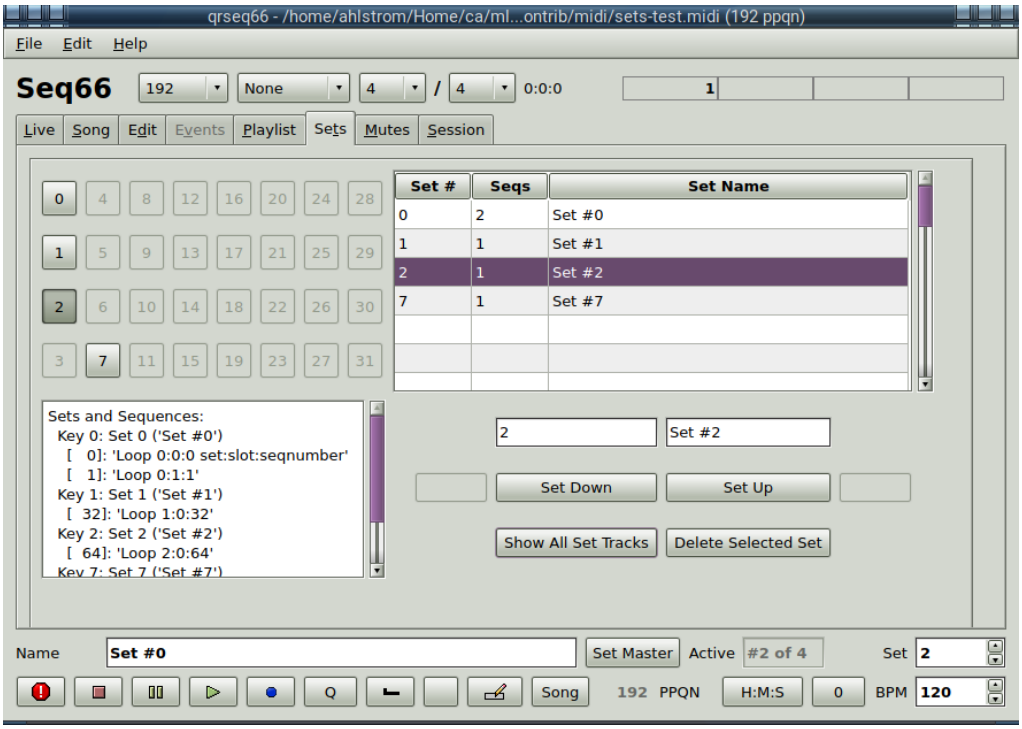


Figure 27: Sets Tab

The operations that can be done consist of viewing the sets, making a screenset active, rearranging the sets, removing sets, and getting a survey of the contents of the sets.

1. Sets Grid. The set grid is (presumably) 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 sets that are present in the MIDI file, and 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)

2. Set List. The table at the right shows the set numbers, how many patterns/sequences are in each set, and the set name. Eventually the set name will be editable.

3. Set Number/Name Fields. Eventually, these fields will be editable.

4. Set Up/Down Buttons. These buttons allow the user to move the sets up and down.

5. Show All Set Tracks. Clicking this button lists all the sets and their tracks in the edit box at the left. This functionality is provisional.

6. Delete Selected Set. Clicking this button deletes the set selected in the table above it. Note that the 0th set cannot be deleted. In *Seq66*, there must always be a set 0.

11.1 Set Handling

This section talks about how sets work. The topics are

- **Set Management**
- **Empty-Set Handling**

11.1.1 Set Management

This section will discuss the work-flows of using sets to organize a song and to control playback.

MORE TO COME.

11.1.2 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 22.2 "Sequencer-Specific Meta-Events Format" on page 162). 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 28: 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.

12 Seq66 Mutes Master

The **Mutes** tab is a way to get a global view of all the mutegroups in a *Seq66* MIDI file or global configuration, and to be able to do some simple operations with the mute groups. To learn more about mute-groups, see section 1.5.3.8 "Mute Group Learn Button" on page 16, and section 21.1.8 "Concepts / Terms / group, mute-group" on page 160.

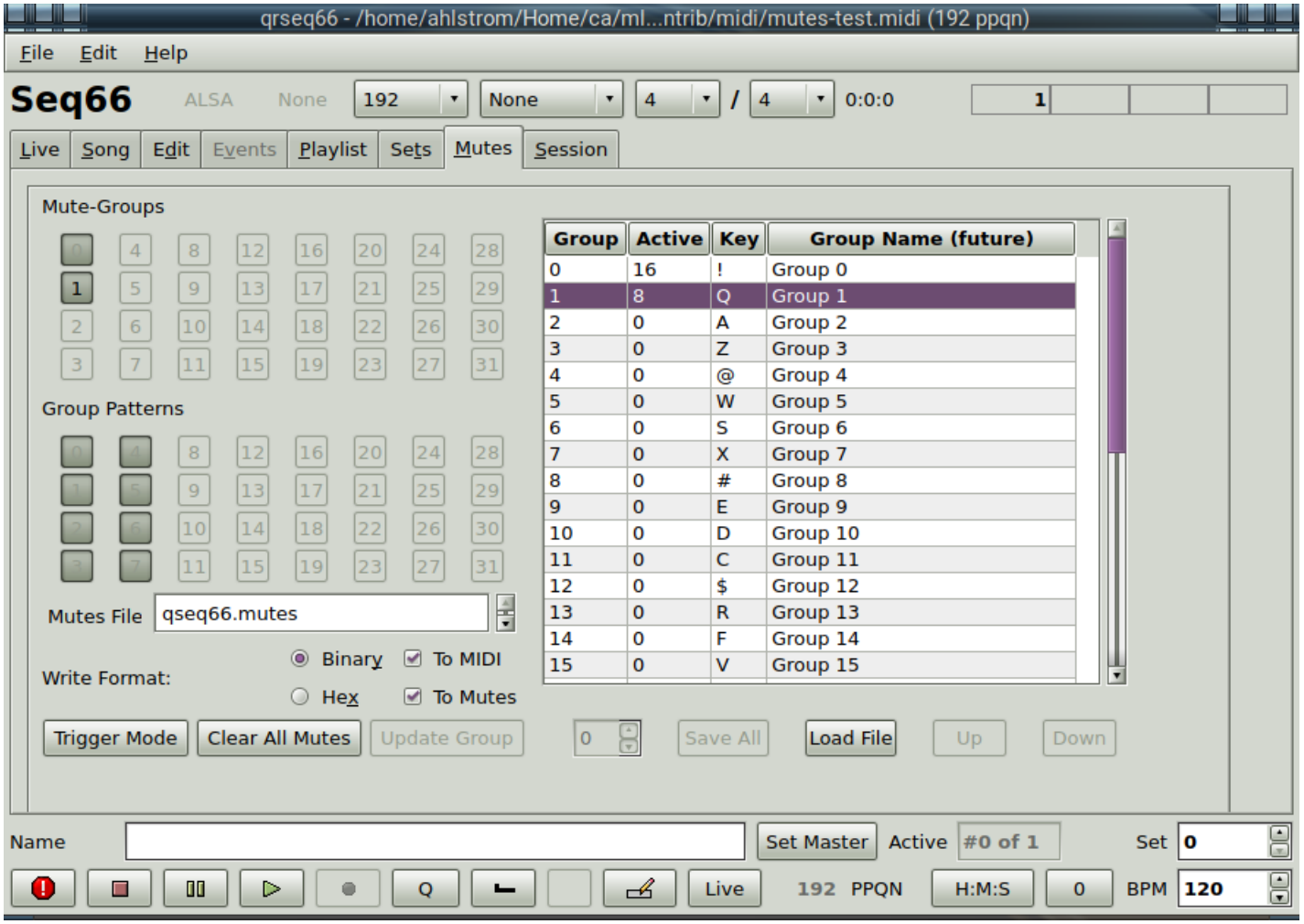


Figure 29: Mutes Tab

This diagram show 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 consists of. This tab make it easy to see the layout of the mute-groups, and also allows for some editing of the mute-groups.

1. Group Table. At the right is a table that holds all of the assigned mute-group key 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 buton 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 put that mute-group in place on the patterns in the current screenset. 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.

Currently, none of the items in this table are editable directly. The user must modify `qseq66.mutes`

with a text editor. This table is the only way to select a mute-group for editing. Click on the desired group, and then click on the group button, perhaps twice, to be able to add pattern mute states via the pattern buttons.

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 7, which now becomes active in the user-interface. (But it is not made active in the patterns panel). The **7** 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, which also enables all of the buttons in the **Group Patterns** grid.

3. Group Patterns. Once this grid is enabled, each button can be click to add a pattern to the mute-group, or remove a pattern from the mute-group.

4. Update Group. When a change in the mute-group status or the status of one of its patterns is made, this button becomes enabled. Once clicked, the current mute-group is modified internally, where it will later be saved when *Seq66* exits, or when the **Save All** button is clicked.

5. Mutes File. The mutes-file shows the base-name of a file into which one can write the current-mute group setup, as a way to back up the setup. TO DO: We need to disable auto-save of the mutes file at exit in this case, unless the name provided is identical.

6. Save All. This button saves all of the mute-groups. It is enabled when a change has been made to a mute-group and has been registered by pressing the **Update Group** button. If the user has provided a path in the **Mutes File** field, the path is stripped. We do not want to write configuration information outside of the session configuration directory. The file is saved, but is not made official in the 'rc' file; one must edit the 'rc' file to use the new 'mutes' file. We might provide a button for that function at some point.

7. Write Format. This section provides the following features, which still need some work:

- **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 normal *Seq66* configuration file (e.g. `qseq66.mutes`).

8. Clear All Mutes. This button will clear every mute group. Use it carefully!

9. Trigger Mode. When activated, this option will enable the **Mute-Groups** buttons, deactive them all, and turn them into standard push-buttons. When clicked the mute-group will be actived during playback.

10. Pattern Offset. If the user has selected a larger set size that is a multiple of 32, this item is enabled. It then allows the user to modify patterns with a sequence number greater than 31. A future feature.

11. Up/Down Buttons. A future feature to move mute-group around without changing the keystroke for that mute group.

13 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 16 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 16 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 *Seq64*.

13.1 Palettes Setup

The palette file is a standard *Seq66* configuration file with a name something like `qseq66.palette`, plus two sections:

```
[palette]
[ui-palette]
```

The first section is the "Pattern" palette, and the second section is the "Ui" palette, which includes the inverse palette as well.

13.1.1 Palettes Setup / Pattern

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" (opqque); we have not yet experimented with changing them. Lastly, only 32 entries are accepted.

13.1.2 Palettes Setup / Ui and Inverse Ui

The following shows the pattern 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 ]
             ...             ...       ...       ...       ...
      13        "Beat Line" [ 0xFF2F4F4F ]  "Beat Line" [ 0xFF2F4F4F ]
      14        "Step Line" [ 0xFF778899 ]  "Step Line" [ 0xFF808080 ]
      15            "Extra" [ 0xFF778899 ]      "Extra" [ 0xFFBD6BB7 ]
```

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 16 entries are accepted.

13.1.3 Palettes Setup / Brushes

The last palette is small, allowing the fill-pattern of a few pattern-editor items to be changed.

```
      [brushes]
      empty = nobrush
      note = solid
      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 used yet. The entry `note` affects the fill of normal/selected notes. The entry `note` 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 `note` affects the fill of the background sequence. The hatching used here helps further distinguish the real notes from the background notes.

13.2 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.

14 Seq66 Keyboard and Mouse Actions

This section presents some tables summarizing keyboard and mouse actions available in *Seq66*. It does not cover mute keys and group keys, which are well described in the keyboard options for the main window. 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.

14.1 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 fields in the **Keyboard** tab:

- Some of the keystrokes are hard-wired, such as "arrow" keys (for controlling play-lists), "page up/down" keys, or the "zoom" 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. 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 key and MIDI control have been consolidated. 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.

A goal of *Seq66* is being able to edit a pattern using mainly the computer keyboard. *Seq66* supports two 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 a snapshot key, change the state of the patterns for live playback, and then release the snapshot key 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 **File** / **Options** / **Keyboard** / **queue**) 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.

One-Shot causes a slot to be queued for only a single playback. The default key is the pipe, `"|"` key. Currently buggy.

12. 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.

13. 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.

14. 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 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 an illegal mute-group keystroke. Also see section 12 "Seq66 Mutes Master" on page 120.

15. **Disable.** It is the inverse **apostrophe** key by default. This key is the *group off* key.
16. **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 **File** / **Options** / **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.

14.2 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 1 "Main Window Support" on page 128.

Table 1: 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 3 "Patterns Panel" on page 31, 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.

14.3 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.

14.3.1 Performance Editor Piano Roll

Note that the keystrokes in this table (see table 2 "Performance Window Piano Roll" on page 129) 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 2: 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 5 "Song Editor" on page 59, 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.

14.3.2 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 3: 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.

14.3.3 Performance Editor Names Section

Table 4: Performance Editor Names Section

Action	**Normal**	**Double**	**Shift**	**Ctrl**
Left-Click	Toggle track	—	Toggle other tracks	—
Middle-Click	—	—	—	—
Right-Click	New/Edit menu	—	—	—

14.4 Pattern Editor Piano Roll Keystrokes

The pattern/sequencer editor piano roll is a complex and powerful event editor; table 5 "Pattern Editor Piano Roll" on page 132, 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.
- `q`. Quantize selected notes. Currently **broken**.
- `t`. Tightened (partial quantize) selected notes. Currently **broken**.
- `r`. Randomize selected notes. Currently **broken**.
- `p`. Enter paint/drawing mode for notes.
- `x`. Exit paint/drawing mode.

14.4.1 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 5: Pattern Editor Piano Roll

Action	**Normal**	**Double**	**Shift**	**Ctrl**
Del	Delete Selected	—	—	—
c	—	—	—	Copy
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 4.4.1 "Pattern Editor / Piano Roll Items" on page 48, 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.

14.4.2 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.

14.4.3 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.

14.4.4 Pattern Editor Virtual Keyboard

Table 6: Pattern Editor Virtual Piano Keyboard

Action	**Normal**	**Double**	**Shift**	**Ctrl**
Left-Click	Play note	—	—	—
Right-Click	Toggle labels	—	—	—

14.5 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.

15 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:

```
C:/Program Files (x86)/Seq66/data/readme.windows
```

Another useful reference is [29].

First, apart from cloning *Sequencer64-packages* (where the *Seq66* installers are kept, which is a **lot** of data), there are tricks to getting the installer (e.g. `seq66_setup_0.95.0.exe`) properly. One can't just right-click and save the link. The file downloaded that way is broken. Instead, click on the link. Then look for a "Download" button, and click that instead.

Installation itself is straightforward. Run the installer (e.g. `seq66_setup_0.93.0.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 is also a `qpseq66-release-package-0.93.0.7z` portable installer than can be download, again using the *GitHub* "Download" button. Just extract that file where desired.)

Note that, although the *Windows* version can be built as a 64-bit application, it is currently built as a 32-bit application.

Now run `C:/Program Files (x86)/Seq66/qpseq66.exe`. (One might want to create a desktop short cut; one can also go to the "Start" menu and search for "qpseq66.exe".) Assuming there are no MIDI devices attached, and no other MIDI programs running, an error like the following will appear:

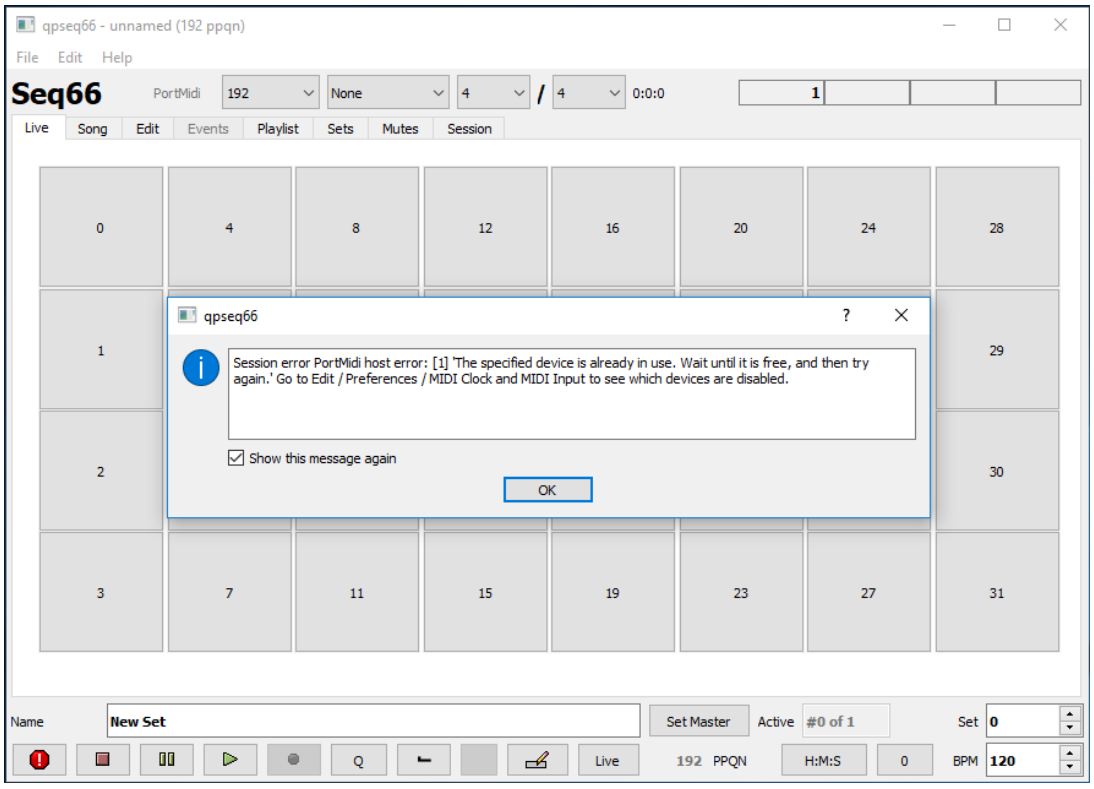


Figure 30: Seq66 First Startup in Windows

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*. It is fixed easily by exiting and rerunning the application: Click **OK** on the error and then **File** / **Quit**. *Seq66* will save this configuration, disabling port 1 and enabling port 0.

Run the *qpseq66.exe* shortcut and load a tune from the directory `C:/Program Files (x86)/Seq66/data/midi`.

Tip: Also note that, in some cases, the application might not play (i.e. it is unresponsive) when it is first started after a new installation. Exit the application and try again; this makes sure that the initial configuration files are created. (At some point we will figure out why this happens).

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:

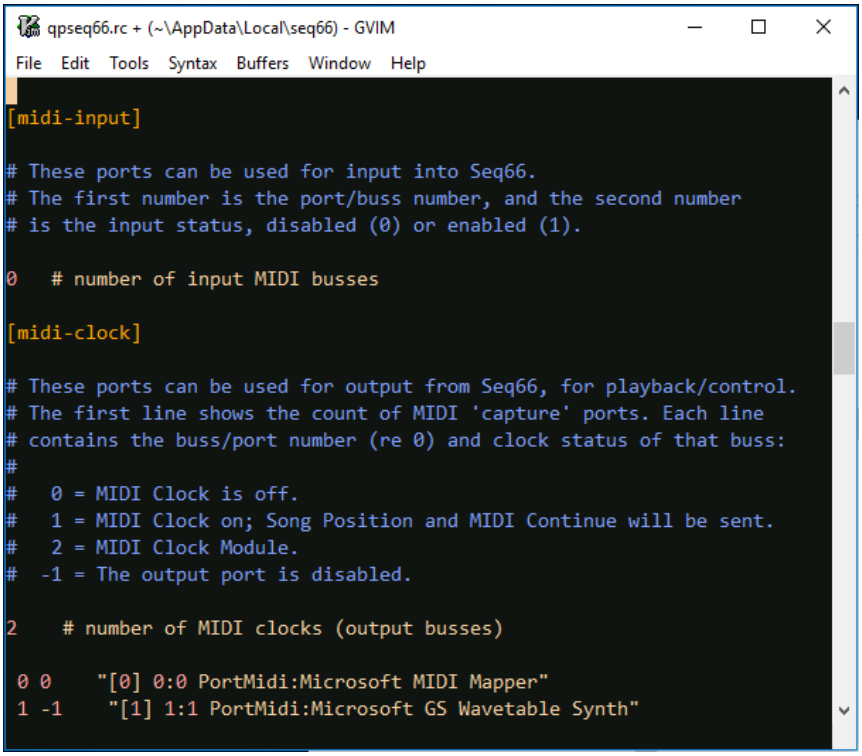


Figure 31: 'rc' File After Exiting First Startup

The `[midi-input]` section indicates there are no input ports (if 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.

Now run `qpseq66.exe` again. No error should appear. Go to **Edit** / **Preferences** / **MIDI Clock**. It might be difficult to click on that tab, and we have never figured out why. Use `Alt-C` if necessary. It will look like this:

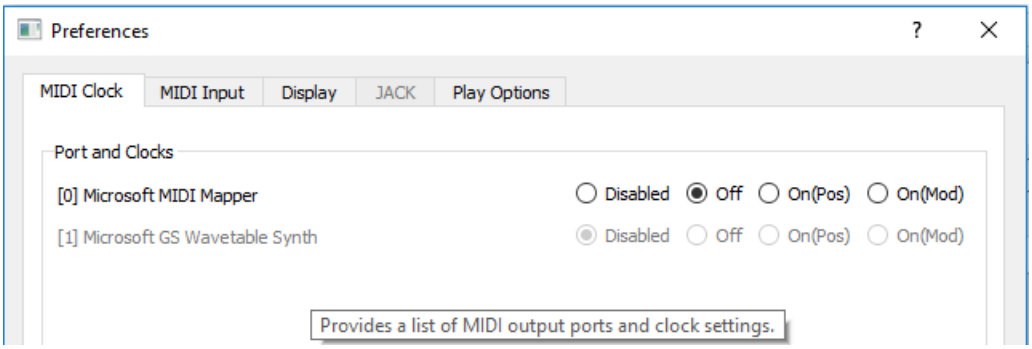


Figure 32: MIDI Output Settings at Second Startup

Next select **File** / **Open** and select this sample tune:

```
C:/Program Files (x86)/Seq66/data/midi/b4uacuse-gm-patchless.midi
```

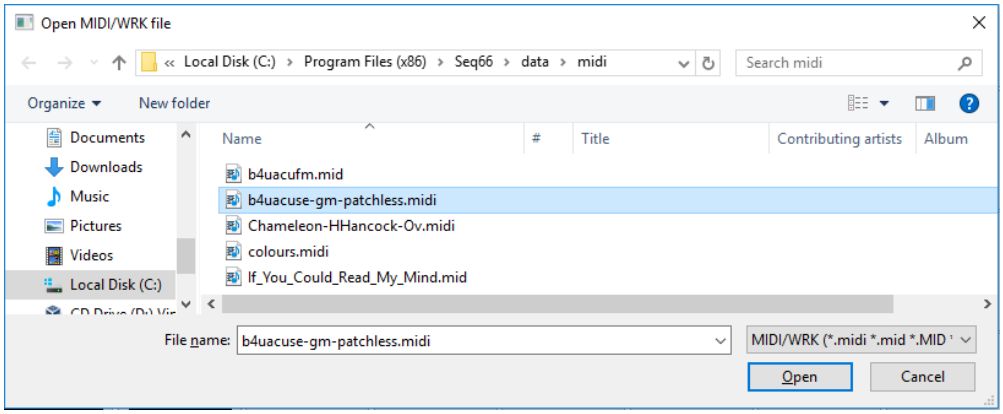


Figure 33: MIDI File Selection

After clicking **Open**, the following set of patterns is shown. Note the two highlighted areas, "Output Selector" and "Song/Live Button".

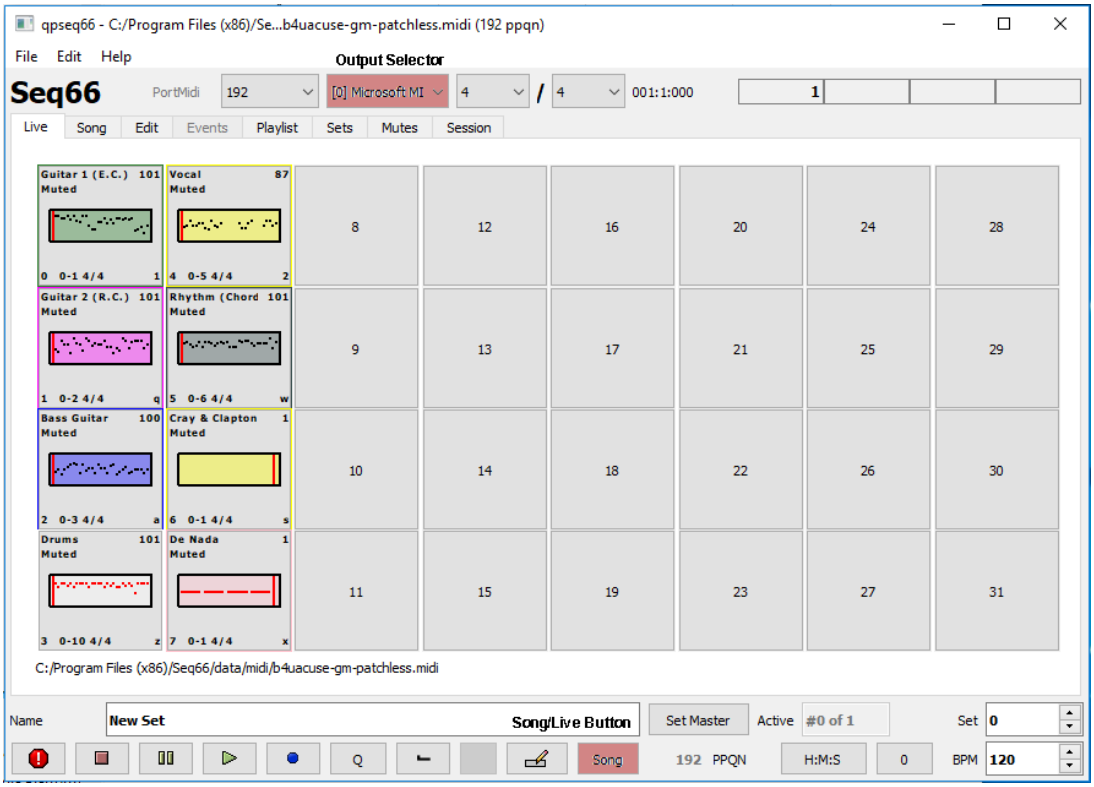


Figure 34: Opened MIDI File

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 (x86)/Seq66/data/readme.windows
```

16 Seq66 ALSA

This section describes some details concerning the ALSA support of *Seq66*. Currently, it just includes some tricks that might be useful.

16.1 Seq66 ALSA / Through Ports

When running the commands `aplaymidi -l` or `arecordmidi -l`, one sees something like:

```
Port    Client name                      Port name
14:0    Midi Through                     Midi Through Port-0
28:0    nanoKEY2                         nanoKEY2 MIDI 1
 . . .
```

The MIDI Through port is useful for...? See [2].

> "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?"

Notice that there is only one Thru port. 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 this 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`.

```
# vi /etc/modprobe.d/alsa-base.conf
options snd-seq-dummy ports=2
```

Save it. No need to reboot, just remove and reinsert the module:

```
# 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.

16.2 Seq66 ALSA / Virtual MIDI Devices

https://tldp.org/HOWTO/MIDI-HOWTO-10.html

> 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 card 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`....

https://freeshell.de/~murks/posts/ALSA_and_JACK_MIDI_explained_(by_a_dummy_for_dummies)/

16.3 Seq66 ALSA / Trouble-Shooting

This section describes some trouble-shooting that can be done with ALSA.

16.3.1 Seq66 ALSA / Trouble-Shooting / MIDI Clock

16.3.1.1 ALSA MIDI Clock Send

To verify that *Seq66* is sending MIDI clock, the easiest way (using ALSA) is to start the following command and set *Seq66* to send MIDI Clock to the `aseqdump` port that appears after restarting *Seq66*:

```
$ aseqdump
```

Once set up, start playback on *Seq66*. The result should be a never-ending stream of rapid MIDI clock messages.

16.3.1.2 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**, check-mark the first "seq24" port. There should be no need to restart.

Now 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!

17 Seq66 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. The JACK support is very loosely based on the RtMIDI project [20]. This mode also supports fallback-to-ALSA if the JACK server is not running.

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`, `-C`/`--jack-master-cond`, and `-t`/`--jack-midi` are available. Also see section 9.3.10 "'rc' File / JACK Transport" on page 92.

If one wants to use *Seq66* and USB devices with JACK MIDI, one needs to expose the USB ports to JACK using `a2jmidid --export-hw`, and connect the resultant MIDI JACK ports oneself, using *QJackCtl*, for example.

The following sections discuss the JACK transport support and the native JACK MIDI support.

17.1 Seq66 JACK / Transport

JACK transport support is *separate* from native JACK MIDI support. 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?

17.2 Seq66 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` or `--jack-midi` option to the command-line. If *Seq66* doesn't find JACK, it will still 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.

17.2.1 Seq66 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.

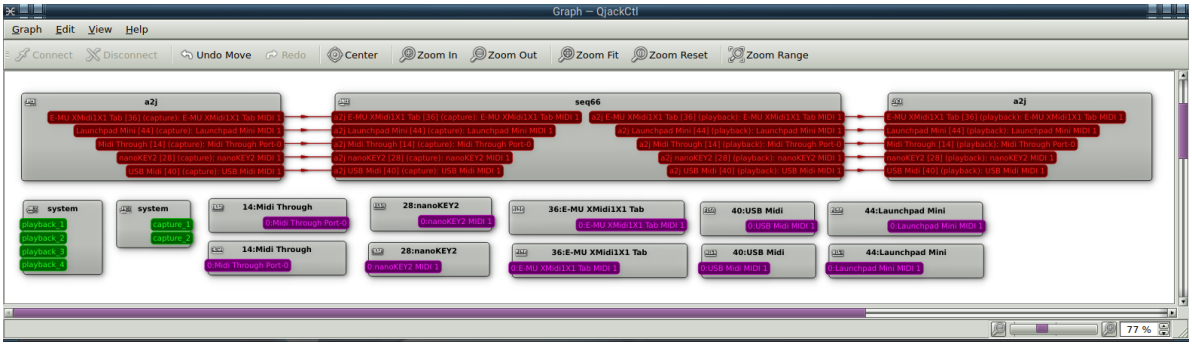


Figure 35: JACK MIDI Ports and Auto-Connect

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 "a2j" 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"
```

17.2.2 Seq66 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`.

17.2.3 Seq66 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*.

17.2.4 Seq66 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.

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.

17.3 Seq66 JACK / Trouble-Shooting

This section describes some trouble-shooting that can be done with JACK.

17.3.1 Seq66 JACK / Trouble-Shooting / MIDI Clock

17.3.1.1 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.

17.3.1.2 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!

17.4 Seq66 JACK / QJackCtl

This section is a placeholder for discussion of *QJackCtl*, its usage of the (deprecated) *JACK Session API*, and its "patchbay" (see [16]).

True?

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.

17.5 Seq66 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`).

What happens on your system?

18 Port Mapping

Seq66, like *Seq24*, bases its I/O port scheme on buss numbers (also called "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 I/O MIDI ports active in the system. In "automatic" (non-virtual, non-manual) mode these ports represent the hardware ports and application ports. In "manual" mode, these ports represent virtual ports that can be connected through other software under *ALSA* or *JACK*.

A given pattern/loop/sequence can be assigned to output to a given port via a buss number that is saved with the pattern. Thus, when a tune is loaded, each sequence can automatically output to the desired MIDI device, by number.

The problem is that the list of MIDI devices can change, with devices being reordered, removed, or added to the set of 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 are too long and change between the systems of each user. Better to let the user determine the mapping of port numbers. Port mapping provides a solution to this issue. It allows the buss number stored with a pattern to be remapped to another buss number, based on the "nick name" of the port. It uses a simple lookup to map names to numbers.

The "nick name" is a shortened version of the port name provided by the MIDI device. 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. Thus, a nick-name `E-MU XMidi1X1` would work.

As with the normal port listings, the port-mappings are managed in the *Seq66* 'rc' file.

18.1 Output Port Mapping

Assume that the system has the following set of ports. These busses are stored in the 'rc' file when *Seq66* exits.

```
[midi-clock]
7      # number of MIDI clocks (output busses)
0 0    "[0] 14:0 Midi Through Port-0"
1 0    "[1] 32:0 nanoKEY2 MIDI 1"
2 0    "[2] 36:0 Launchpad Mini MIDI 1"
3 0    "[3] 40:0 Q25 MIDI 1"
4 0    "[4] 36:0 E-MU XMidi1X1 Tab MIDI 1"
5 0    "[5] 128:0 yoshimi:input"
6 0    "[6] 128:0 FLUID Synth (1070760):Synth input port (1070760:0)"
```

If some items are unplugged, then this list will change, so we save it while still running *Seq66*: click the **Save Clock/Input 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   "Midi Through Port-0"
1   "Launchpad Mini MIDI 1"
2   "nanoKEY2 MIDI 1"
3   "yoshimi:input"
4   "E-MU XMidi1X1 Tab MIDI 1"
5   "FLUID Synth"               # "qsynth midi_00"
```

It is simpler, containing only an index number and shorter versions of the port names, called "nick-names". These index numbers can be used like buss numbers: they can be stored in a pattern, and used to direct output to a device by name. Let's say we've unplugged some devices, so that the system MIDI clocks list is now shorter:

```
[midi-clock]
4      # number of MIDI clocks (output busses)
0 0    "[0] 14:0 Midi Through Port-0"
1 0    "[4] 32:0 E-MU XMidi1X1 Tab MIDI 1"
2 0    "[2] 36:0 Launchpad Mini MIDI 1"
3 0    "[3] 128:0 yoshimi:input"
```

So, if a pattern has stored mapped-buss 2 "E-MU XMidi1X1 MIDI 1" as its output buss, and the output port map is active, the "2" is looked up in the map, the nick-name "E-MU XMidi1X1 MIDI 1" grabbed, looked up in the system list, which returns "4" as the actual system buss number to use for output.

On the other hand, 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 rerouted 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.

Also note the entries for "FLUID Synth". The long name is really long, and it contains a semi-random numerical ID that changes every time QSynth is run. For simplicity, only the name before the parenthesis is saved as the nick-name. Fortunately, only one instance of Qsynth can run. However, note that "FLUID Synth" is the name of *QSynth* under *ALSA*, while "qsynth midi_00" is the name under *JACK*.

If the map is not active, then only the actual system output ports are shown in the user interface.

18.2 Input Port Mapping

The input ports are handling somewhat similarly. Here's another initial system input setup:

```
[midi-input]
6      # number of input 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] 36:0 E-MU XMidi1X1 Tab MIDI 1"
4 0    "[4] 40:0 USB Midi MIDI 1"
5 0    "[5] 44:0 Launchpad Mini MIDI 1"
```

Note that the "system:announce" buss is always disabled, as *Seq66* does not use it. Here is the stored input port-map:

```
[midi-input-map]
0   "announce"
1   "Midi Through Port-0"
2   "nanoKEY2 MIDI 1"
3   "E-MU XMidi1X1 Tab MIDI 1"
4   "USB Midi MIDI 1"
5   "Launchpad Mini MIDI 1"
```

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.

18.3 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   "announce"
1   "Midi Through Port-0"
2   "Launchpad Mini MIDI 1"
3   "nanoKEY2 MIDI 1"
4   "Q25 MIDI 1"
5   "E-MU XMidi1X1 Tab MIDI 1"


[midi-clock-map]
1   # map is active
0   "Midi Through Port-0"
1   "Launchpad Mini MIDI 1"
2   "nanoKEY2 MIDI 1"
3   "Q25 MIDI 1"
4   "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.

Perhaps tricky, but once one has set up a whole suite of I/O device maps, one can reliably use these hard-wired 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 Mini.

19 Seq66 Headless Version

Seq66 can be built as a command-line application. See the `INSTALL` file provided with the source code distribution. That is, *Seq66* can be run from the command-line, with no visible user interface. It can also be instantiated as a Linux daemon, for totally headless usage. Because there is not a lot of visibility into a headless process, the setup for `seq66cli` is a little complex, and the musician must get used to blind MIDI control.

19.1 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 input buss must be *enabled* by setting the active bit to "1" for that device.
3. The MIDI 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.
4. 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.
5. The desired outpuit buss must be *enabled* by setting the active bit to "1" for that device.
6. Use the desired play-list. The headless sequencer can only select songs to play via a pre-configured play-list.

Once the above steps are proven for the `qseq66` configuration files, then the same settings can be made for the `seq66cli` configuration files.

Sometimes odd problems, such as the output synthesizer not working, not appearing in the list of outputs, can prove 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 20 "Launchpad Mini" on page 151.)

Second, start the desired (software) synthesizer. We will 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. 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 the program, exit it, and see the results in the '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 'rc' file to be read. The `nanomap.ctrl` file sets up the *nanoKEY2* as shown in this figure:

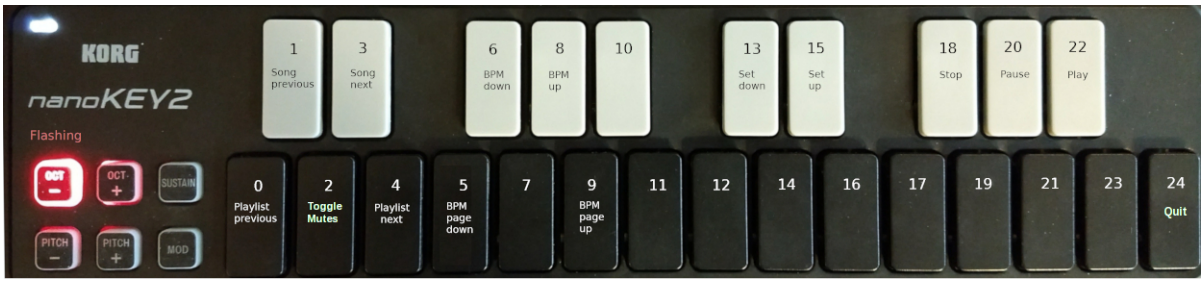


Figure 36: Sample nanoKEY2 Control Setup

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]
"nanomap.ctrl"
```

The `seq66cli.rc` file should also enabled 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:

```
load-key-controls = true
load-midi-controls = true
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. Run `seq66cli` and then stop it (Ctrl-C) to generate the initial configuration files, `$HOME/.config/seq66/seq6`
2. Copy the contents of of `data/seq66cli/` to `$HOME/.config/seq66`.
3. Copy `data/samples/sample.playlist` and `data/samples/sample.playlist` to `$HOME/.config/seq66`.
4. 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 or 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 run `seq66cli` as a daemon:

```
$ seq66cli --buss 2 --verbose -o daemonize --playlist data/sample.playlist
```

The keyboard controls and sound output should work. However, at present the daemon doesn't get the proper settings, so that is something to work on for version 0.95. For now, create a shortcut to run the application in the background. In a *Fluxbox* menu:

```
[exec] (Seq66 Headless) {/usr/bin/seq66cli --jack-midi --jack-master --bus 6}
```

For a more sophisticated setup, see section 20 "Launchpad Mini" on page 151.

20 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).

20.1 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):

```
       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
```

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*.

20.2 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.

20.3 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]
"qseq66.ctrl"
```

to

```
[midi-control-file]
"qseq66-lp-mini.ctrl"
```

In `qseq66-lp-mini.ctrl`, first read through the file to get familiar with the format and purpose of this file.

20.3.1 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]
control-buss = 0xff     # change 0xff to 2
```

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.

There are three sets of controls: loops, mute-groups, and automation, as described in the following sections.

20.3.1.1 [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 first four 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"  [ 0x90  0 1 127 ] [ 0x90  0 1 127 ] ...
 1 "q"  [ 0x90 16 1 127 ] [ 0x90 16 1 127 ] ...
 2 "a"  [ 0x90 32 1 127 ] [ 0x90 32 1 127 ] ...
 3 "z"  [ 0x90 48 1 127 ] [ 0x90 48 1 127 ] ...
```

The note values (0, 16, 32, 48) are in decimal. Why? Less to type. The whole section is 32 lines, so only the top 4 rows of the Mini are assigned to loop-control by this configuration file. The pattern numbers are shown inside 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 and being muted.

20.3.1.2 [mute-group-control]

The mute-group controls are similar, except we didn't bother filling the On and Off stanzas at this time; they are all zeroes.

```
[mute-group-control]
 0 "!"  [ 0x90  64 1 127 ] ...
 1 "Q"  [ 0x90  80 1 127 ] ...
 2 "A"  [ 0x90  96 1 127 ] ...
 3 "Z"  [ 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).

20.3.1.3 [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.

```
0 "'" [ 0x00 0 0 0 ] [ 0xb0 104 127 127 ] [ 0xb0 104 127 127 ] # BPM Up
1 ";" [ 0x00 0 0 0 ] [ 0xb0 105 127 127 ] [ 0xb0 105 127 127 ] # BPM Dn
2 "]" [ 0x00 0 0 0 ] [ 0xb0   0   0   0 ] [ 0xb0   0   0   0 ] # Set Up
3 "[" [ 0x00 0 0 0 ] [ 0xb0   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.

20.3.2 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). Change

```
[midi-control-out-settings]
output-buss = 0xff
midi-enabled = false
```

to

```
[midi-control-out-settings]
output-buss = 1
midi-enabled = true
```

If the device starts lighting up mysteriousy while playback is happen, make sure the music is not being played to the control device channe. 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! ☺)

20.3.2.1 [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.

```
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.
- **Armed**. This stanza is configured to show a queued pattern slots as yellow.
- **Armed**. This stanza is configured to show an empty pattern slots as off (dark).

20.3.2.2 [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:

```
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 ]
```

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.

20.3.2.3 [automation-control-out]

This section allows for the following status to be shown in the top row of circular buttons:

```
1 [ 0xb0 104 60 ] [ 0xb0 104 0 ] # panic
1 [ 0xb0 104 60 ] [ 0xb0 104 0 ] # play on/off
1 [ 0xb0 104 15 ] [ 0xb0 104 0 ] # stop on/off
1 [ 0xb0 104 62 ] [ 0xb0 104 0 ] # pause on/off
```

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.

20.4 Test Run, ALSA

Now that we're set up, start *Seq66*.

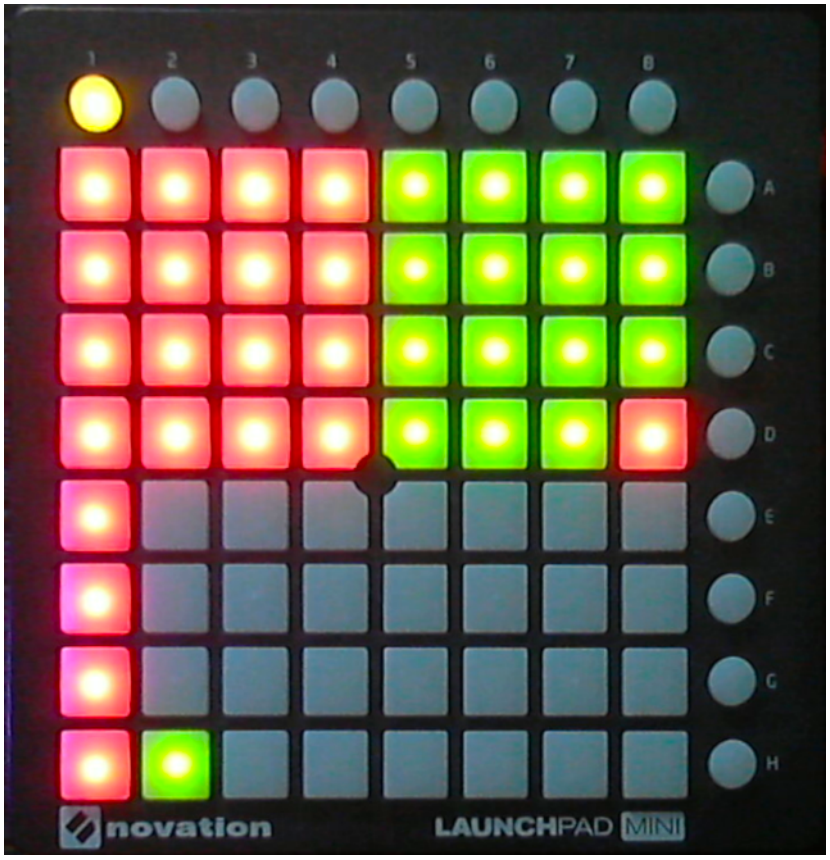


Figure 37: Launchpad Minu Running with Seq66

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 pictre.

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 product a light show!

20.5 System Survey, JACK

Now let's see what we have to do in *JACK*. First peruse section 17 "Seq66 JACK" on page 139, to understand the basics about *JACK*, including the last section there that describes how to set up *ALSA*-to-*JACK* bridging.

Run the following command, verify the ports in **Edit** / **Preferences** / **MIDI Clock** and **MIDI Input**, and then exit.

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

In `qseq66.rc`, one will find this setting:

```
1     # with_jack_midi
```

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 20.4 "Test Run, ALSA" on page 157.

21 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.

21.1 Concepts / Terms

This section doesn't provide comprehensive coverage of terms. It covers terms that might be puzzling.

21.1.1 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.

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 loo 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.

21.1.2 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". Performing an *arm* operation in *Seq66* means clicking on an "unarmed" sequence in the patterns panel (the main window of *Seq66*). An unarmed sequence will not be heard, and it has a normal background. 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.
- 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.

21.1.3 Concepts / Terms / bank, screenset, play-screen

The *screen set* 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. For the most part, the column number is best left at 8. The term "bank" is *Kepler34*'s name for "screen set".

By default, only one set is active and playing at a time. This set is informally termed the "play screen".

21.1.4 Concepts / Terms / buss, bus, port

A *buss* (also spelled "bus" these days; 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.

21.1.5 Concepts / Terms / performance, song, trigger

In the jargon of *Seq66*, a *performance* or *song*is an organized collection of patterns that play a 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.

21.1.6 Concepts / Terms / Auto-step, Step-Edit

Auto-step (step-edit) provide 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.

21.1.7 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.

21.1.8 Concepts / Terms / group, mute-group

A *group* in *Seq66* is a set of patterns, that can arm (unmute) their playing state 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. When applied, the mute-group enables one or more patterns in the current screenset.

21.1.9 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 9.3 "'rc' File" on page 89) 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.

21.1.10 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. Check out the source code.

21.1.11 Concepts / Terms / queue, keep queue, snapshot, one-shot

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 end 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.)

21.2 Concepts / Sound Subsystems

21.2.1 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].

21.2.2 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 [17] for the PortMIDI home page; our version cuts out code that requires *Java*.

21.2.3 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 [5].

22 MIDI Format and Other MIDI Notes

Seq66 tries to handle the most common 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. (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`

22.1 Standard MIDI Format 0

Seq66 can read and import SMF 0 MIDI files, and will split 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 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*.

22.2 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 or implement it.

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" 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. The format of sequencer-specific meta-events is:

```
FF 7F len data
```

The first byte of the data is the "manufacturer ID", which *Seq24* set to "24". Another "24" is added to make the number `0x2424` easy to search in a binary hex editor, such as *hexer*. 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 format:

```
FF 7F len 24 24 00 nn databyte(s)
```

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. (For now).

All of the `SeqSpec` s 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` section appear only if they are non-default. 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 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()`.

22.2.1 SeqSpec c_midibus

`c_midibus`: specifies the desired output buss/port number for a track.
Length: 5
Format: `0x24240001 buss`

The 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 18 "Port Mapping" on page 145.

Table 7: 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_reserved_1	TBD	0	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	To-do	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...

22.2.2 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 held by an 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 support for "channel" 0x80 (a flag to use the events channels) is new and

still being worked out.

22.2.3 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 9.3.6 "'rc' File / MIDI-Clock Section" on page 91.

22.2.4 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.

22.2.5 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 11 "Seq66 Set Master" on page 118.

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.

22.2.6 SeqSpec c_timesig

`c_timesig`: specifies the time signature for a track.
Length: 6
Format: `0x24240006 bpb bw`

This `SeqSpec` specifies the 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. However, in addtion to being set for the sequence, it is also set globally in the `seq66::performer` class, and the last one encountered is official. Bug, or feature?

The normal time-signature meta-event is 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.

22.2.7 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()`.

22.2.8 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.

22.2.9 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 background 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.

22.2.10 SeqSpec c_mutegroups

`c_mutegroups`: specifies the mute-groups in play in a song.
Length: 4 + 4 * groupcount + 4 * seqcount
Format: `0x24240009 groupcount:S seqcount:S groupdata...`

Mute-groups are a feature that enable the toggling of arming/muting for multiple patterns at once. *Seq66* supports up to 32 mute-groups of size 32 patterns. More flexibility is planned and partially supported now.

Mute-groups can also be stored in a 'mutes' files, which is probably a better place to store them. (See section 9.6 "'mutes' File" on page 111, and section 29 "Mutes Tab" on page 121). 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 it is recommended to use only the 'mutes' file, unless the song absolutely needs its own set of mute-groups.

22.2.11 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.

22.2.12 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.

22.2.13 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.

22.2.14 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.

22.2.15 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.

22.2.16 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 9.7 "'drums' File" on page 112). 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 9.3.4 "'rc' File / Note Mapper" on page 90. 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.

22.2.17 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.

22.2.18 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.

22.2.19 SeqSpec c_tempo_map

`c_tempo_map`: Not implemented.
Length: 5
Format: `0x24240017`

This section is an unimplemented *Seq32* feature.

22.2.20 SeqSpec c_reserved_(1_2)

`c_reserved_1_2`: Reserved for expansion.
Length: Indeterminate
Format: `0x24240018, 19`

22.2.21 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.

22.2.22 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 13 "Palettes for Coloring" on page 123. 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).

22.2.23 SeqSpec c_seq_edit_mode

`c_seq_edit_mode`: specifies the editing mode of each pattern.
Length: Indeterminate
Format: `0x2424001C`

This looks like a way to specify that a pattern should be edited in drum mote (1) or normal mode (0). This is a *Kepler34* feature. However, in *Seq66* it is currently not implemented. We don't even read it.

22.2.24 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 future feature to provide a user's request for a one-shot pattern, a pattern that plays once 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. Not yet read nor written. However, a loop-count is used in the one-shot step-edit feature used in recording a device's drum pattern. See section 4.7 "Pattern Editor / Bottom Row" on page 54.

22.3 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.

22.3.1 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()`.

22.3.2 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.

22.3.3 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.

`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.

Table 8: MIDI Meta Event Types

Type	Event	Seq66 Handling
0x00	Sequence number	R/W
0x01	Text event	Skipped
0x02	Copyright notice	Skipped
0x03	Sequence or track name	R/W
0x04	Instrument name	Skipped
0x05	Lyric text	Skipped
0x06	Marker text	Skipped
0x07	Cue point	Skipped
0x08-0x0F	Other text events	Skipped
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	Read
0x7F	Sequencer-Specific event	Seq66 data handled

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.
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

To do: 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, they 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.

Table 9: 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

22.3.4 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.

22.3.5 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.

22.3.6 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.

22.3.7 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.

22.3.8 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.

22.3.9 SysEx Event (0xF0)

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

Seq66 has some code in place to store these messages, but the data is currently not actually stored or used. Although there is some infrastructure to support storing the SysEx event within a sequence, the SysEx information is simply skipped. *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 skipped as well.

In *Seq24*, these events are placed at the end of the song, but are not marked as `SeqSpec` data. Most MIDI applications handle this situation fine, but some (e.g. midicvt) do not. *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.

22.3.10 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.

22.4 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.

22.4.1 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.

22.4.2 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.

22.4.3 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.

22.4.4 SeqSpec 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.

23 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.

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*. [8].
- *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 (we hope) of *Seq24*. It preserves (we hope) the lean nature of *Seq24*, while adding many useful features. Without *Seq24* and its authors, *Sequencer64* and then *Seq66* would never have come into being.

24 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.

25 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] Jay Capela Music. *"Combine": A Seq24 Demonstration.* https://www.youtube.com/watch?v=fUiXbVTObJQ. 2010.

[4] Various. *Forks of Seq66.* https://github.com/sivecj/seq66. 2021.

[5] JACK team. *JACK Audio Connection Kit.* http://jackaudio.org/. 2015.

[6] Oli Kester. *Kepler34: Seq24 for the 2010s.* https://github.com/oli-kester/kepler34. 2010-2016.

[7] Linux Audio Users. *Unofficial Linux audio users IRC channel.* http://kiwiirc.com/nextclient/#irc://irc.freenode.net/#lau. 2021.

[8] 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.

[9] Author unknown. *A description of Seq66*. https://libreav.org/software/seq66. 2019.

[10] Coolsoft. *Coolsoft MIDIMapper (Windows)*. https://coolsoft.altervista.org/en/midimapper. 2018.

[11] Coolsoft. *Coolsoft VirtualMIDISynth (Windows)*. https://coolsoft.altervista.org/en/virtualmidisynth. 2018.

[12] linuxaudio.org. *seq24: toggle sequences with a MIDI controller*. http://wiki.linuxaudio.org/wiki/seq24togglemiditutorial. 2013.

[13] midi.org. *Summary of MIDI Messages*. https://www.midi.org/specifications/item/table-1-summary-of-midi-message#2. Year unknown.

[14] Chris Ahlstrom. *Extension of midicomp/midi2text to convert between MIDI and ASCII text format*. https://github.com/ahlstromcj/midicvt. 2015-2016.

[15] 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.

[16] Simon W. Fielding. *QjackCtl and the Patchbay*. https://www.rncbc.org/drupal/node/76. 2008.

[17] PortMedia team. *Platform Independent Library for MIDI I/O*. http://portmedia.sourceforge.net/. 2010.

[18] Benjamin Robert Tubb *MIDI PPQN Duration Calculator*. https://demonstrations.wolfram.com/MIDIPPQNDurationCalculator/. 2011.

[19] *QjackCtl*. https://sourceforge.net/projects/qjackctl/. https://qjackctl.sourceforge.io/qjackctl-index.html. https://git.code.sf.net/p/qjackctl/code qjackctl-git. 2021.

[20] Gary P. Scavone. *The RtMIDI Tutorial*. https://www.music.mcgill.ca/~gary/rtmidi/. 2016.

[21] Seq24 Team. *The home site for the Seq24 looping sequencer*. http://www.filter24.org/seq24/download.html. 2010.

[22] Seq24 Team. *The home site for the Seq66 looping sequencer*. https://edge/launchpad.net/seq24. 2016.

[23] Excds. *A simple mapping for toggling the LEDs on the Novation launchpad together with seq24*. https://github.com/Excds/seq24-launchpad-mapper. 2013.

[24] Stan Preston (stazed). *The home site for the Seq32 looping sequencer*. https://github.com/Stazed/seq32. 2016.

[25] Chris Ahlstrom. *A reboot of the Seq24 project as "Seq66"*. https://github.com/ahlstromcj/seq66/. 2015-2017.

[26] Unknown. *Timing accuracy and resolution*. http://midi.teragonaudio.com/tutr/seqtime.htm. 2000?.

[27] Timidity++ Team. *Download site for Timidity++ source code.* http://sourceforge.net/projects/timidity/. 2015.

[28] VMPK Team. *Virtual MIDI Piano Keyboard.* http://vmpk.sourceforge.net/. 2015.

[29] Donya Quick *Working with MIDI on Windows* http://donyaquick.com/midi-on-windows/#x1-120004.1. 2021.

[30] The Wootangent man. *Seq66 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.

Index