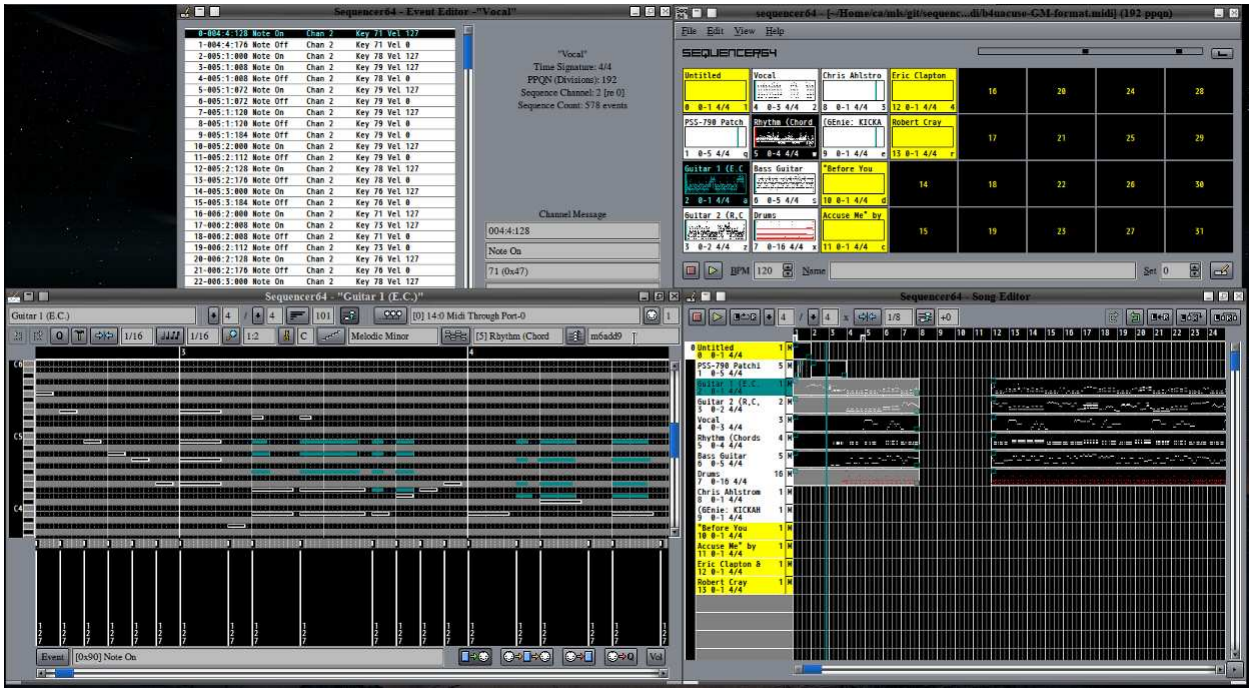
Sequencer64 User Manual 0.90.5

Chris Ahlstrom
(ahlstromcj@gmail.com)

May 7, 2017



Inverse Video Mode

Contents

List of Figures

List of Tables

1 Introduction

This document describes how to use *Sequencer64* [24], through version 0.90.5, using the latest commit from the **master** branch. If one is impatient to get started mastering *Sequencer64*, then proceed to section 1.5 "Let's Get Started!" on page 11.

Sequencer64 has added native JACK MIDI support, and it fixes some bugs in JACK transport, MIDI clocking, and JACK Master mode (with code from the *SooperLooper* and *Seq32* projects). However, as user *stazed* notes on GitHub, there is more that can be done (and he has done much with his *seq32* [21] project).

We've added a many features to make song editing faster. There is now a Pause button/keystroke. More editing and navigation can be done using keystrokes. The armed/mute status of all of the *other* active patterns can be toggled. We've got zoom keys for the editors, current-sequence highlighting, important bug fixes, some optimization, and much more.

We have many contributors to acknowledge. Please see section 14 "Kudos" on page 155.

1.1 Sequencer64: What?

Sequencer64 is an continuation/reboot of *Seq24*, a live-looping sequencer with an interface more like a hardware sequencer than the typical software MIDI sequencer. *Sequencer64* has many updates and improvements, and a modestly new look, so that a new manual is a necessity.

Seq24 was a very active project, with a number of contributors, who created patches, additional functionality, and even ports to Windows. We searched for these updates, and incorporated them into *Sequencer64* where feasible. There are still some things to be done, including a port to Windows.

Sequencer64 is not a synthesizer. It requires a hardware synthesizer, or a software synthesizer such as Timidity [28], FluidSynth [4], ZynAddSubFX [36], Yoshimi [32] [33], etc. See [7] for a long list of "Linux" synthesizers).

Sequencer64, like *Seq24*, works a bit like an *Alesis SR16* drum machine, which, for some, is a very intuitive and fast way to do MIDI. If one has worked with trackers like *SoundTracker* and *ShakeTracker*, then "you are a tracker guy and it gonna go fast". With *Sequencer64*, one creates several patterns, and then combines them.

Unfortunately, *Seq24* spent some time in limbo (2010 was the last major update, with a recent update to fix a MIDI clock issue). There are a number of forks for it on *GitHub*, and some conversions to code in other languages, and some patches. There is also a fairly extensive port to Windows. So, why would we bother creating yet another fork of *Seq24*?

1.2 Sequencer64: Why?

The first reason to reboot *Seq24* is consolidation of many of the features and fixes that it has accumulated in various forks over the years. Also, although "feature-complete", additional features would be useful, such as support for modern session managers, more use of keystrokes, more comprehensive undo/redo, bug fixes, improving the "user" configuration file, and a way to edit parts of the MIDI file textually.

Secondly, we view *Seq24* as a kind of "vi" of MIDI editing... spare, lean, and powerful. It deserves to be a living project. Without *Seq24* and its authors, *Sequencer64* would never have come into being.

1.3 Improvements

The following improvements have been made to *Seq24* for *Sequencer64*.

- The newest update is native JACK MIDI support! At last!

- A new *event editor* dialog provides viewing and editing of MIDI events as text. This editor is basic, but useful.
- *Sequencer64* now reads SMF 0 MIDI files, splitting the file into patterns based on channel number.
- A "Tap" button sets the BPM by tapping a button or a key.
- Some features have been ported from *Stazed*'s *Seq32* [21] project. Some features can be disabled at configure/build time, if desired.
  - The chord-generation feature of *Seq32*.
  - The pattern-transposing feature.
  - MIDI song export. It allows a song/performance to be exported into a MIDI file without *Sequencer64* loops, so that the performance can be played in a standard MIDI sequencer.
  - The extended undo/redo support of *Seq32* has been ported to *Sequencer64*.
  - The ability to split triggers at the nearest snap, instead simply splitting them in half.
  - Editing the amplitude of data events using a simple low-frequency oscillator (LFO).
  - A large number of other *Seq32* features have been ported, but they are currently commented out until we can test them and make sure they are "perfect".
    - Enhanced JACK transport support.
    - Diverting multi-channel recorded MIDI so that each channel is stored in the sequence/pattern configured for that channel
    - Randomizing MIDI events.
    - Selection extensions such as selecting odd/even notes, and handles on data events.
    - More feature for controlling transport.
- An optional second song editor window, to have a view of two different parts of a large song while arranging it.
- Set Tempo and Time Signature events are now processed, incorporated into the user-interface, and are saved as standard MIDI data.
- The scale, key, and background sequence settings are now saved to the *Sequencer64* MIDI file, either globally or per-sequence.
- In the sequence/pattern and song editors, the piano roll now scrolls to keep up with the progress bar for patterns longer than the width of the sequence editor window.
- Additional mouse and keystroke support in the pattern and song editor windows:
  - Usage of Page Up, Page Down, and Shift Page Up, Shift Page Down, to move vertically and horizontally in the pattern and song windows.
  - Moving selected notes or triggers using the arrow keys.
  - Pasting selected notes (and other events) using the arrow and Enter keys.
  - Additional keystroke support for entering and exiting "paint" mode.
  - Usage of the Mod4 (Super, Windows) key to keep editing (painting) mode enabled after releasing the right mouse button; useful with some "modern" crappy uni-touchpads.
  - Shift-left-click on a pattern slot (pattern editor), or on the pattern name or M (mute) (song editor) to toggle the status of all of the other active slots. Useful for listening to a single track by itself.
  - Right-click on a the virtual keyboard in the pattern editor toggles between showing letters/octaves (e.g. "C4") versus the MIDI note numbers.
- The "user" configuration is now written to disk, and settings are added to it as time goes on. It currently offers the long-standing feature of customizable buss, instrument, and controller information, plus some customizations of the user-interface, such as font, display of the main window's pattern grid, colored and thicker progress bars, and modification of the default 40 millisecond window redraw rate.

- Support for mapping MIDI events to a single MIDI buss for testing or simple use cases.
- On-going support for handling PPQN values other than the default value of 192. Still not comprehensive, but very usable.
- Small improvements in appearance:
  - Support for showing empty (having only meta events) sequences in a highlight color (yellow).
  - Support for highlighting the currently-focussed pattern editor window, in black-on-cyan if not armed, and in cyan-on-black if armed. The highlighting appears in both the pattern and song editors.
  - Sequences that are shorter than a quarter note are now padded to one full measure, for smoother scrolling on the patterns panel.
  - Modification of the colors of the scale and background sequence in the sequence editor to make it easier to see them all.
  - A new font, enabled at run time, that is bolder and has a more modern, anti-aliased look.
  - Clean, solid lines to replace the dotted lines in the piano-roll grids.
  - Additional zoom values have been added to support the display of high PPQN sequences.
  - An "inverse" or "night" color mode has been added for those who find the glare of all-white windows to be uncomfortable.
- Consolidated various patches in forks of the *Seq24* project found by searching the web. Fixes to bugs found while refactoring *Seq24* were also made. These fixes are noted in detail in the project's ROADMAP and contrib/notes/bugs-to-investigate files.
- More musical scales (harmonic minor, melodic minor, whole tone, etc.) have been added.
- A pause feature has been added to ALSA-mode playback. It also includes a pause keystroke (".") and a pause button.
- Internal improvements.
  - Provided a new, more MIDI-compliant output format for the MIDI files. The old format can still be read, and, with a "legacy" option, be written.
  - Changed the handling of the MIDI event container, which greatly speeds up the loading of a MIDI file, especially in debug mode.
  - Note-transpose now also works on aftertouch events.
  - Non-note events are now copied, moved, or pasted, even if not visible, in the pattern editor.
  - The code was reformatted using *astyle* and personal preferences, and refactored into smaller modules.
  - Much documentation was added to the code as we figured out how it worked. Generation of Doxygen output (including a PDF file) provides a developer's reference manual.
  - Debian packaging was incorporated into the project to make it easier to install without source code. Bootstrapping and packing scripts were added so that other developers can rebuild the project from scratch.
- New minor features.
  - The size of the Patterns Panel (main window) is now locked.
  - Support for LASH is a run-time or a build option.
  - Support for reading and writing configuration files from the user's `$HOME/.config/sequencer64` directory, or, optionally, other directories.

In the future, version 1.0 will be even more object-oriented, hopefully faster, and easier to modify. Eventually, we might get it to build for Windows, using MingW, though this is a low priority and a fairly significant task. We've also been asked to incorporate support for native JACK MIDI (**Done!**), a scripting language, for OSC (Open Sound Control), and for NSM (the Non Session Manager).

One more note. *Sequencer64* is extremely customizable, and it features can be configured by defining macro names in the source code, by enabling/disabling options at build-configuration time, and by many new 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.

1.4 Document Structure

The structure of this document follows the user-interface of *Sequencer64*. The sections are provided in the order their contents appear in the user interface of *Sequencer64*. To help the reader jump around this document, it provides multiple links, references, and index entries.

Usage tips for each of the functions provided in *Sequencer64* are sprinkled throughout this document. Each tip occurs in a section beginning with "**Tip:**". Each tip is provided with an entry in the index, under the main topic "tips".

Bug notes for some of the oddities found in *Sequencer64* are sprinkled throughout this document. Each bug occurs in a sentence beginning with "**Bug:**". Each bug is provided with an entry in the index, under the main topic "bugs". These bugs are items that we will try to fix as time goes on.

"To-do" items are also present, again in the same vein. Each to-do occurs in a sentence beginning with "**TODO:**". This document currently has a lot of them!

"New" items are also present, in the same vein. New features (post version 0.9.2) will be noted with the tag "**New:**".

1.5 Let's Get Started!

Let us run *Sequencer64*, but run it without using *JACK*, which complicates the discussion of *Sequencer64*. The first thing to do is make sure one has no other sound application running (unless one wants to risk blocking *Sequencer64* or hearing two sounds simultaneously, depending on one's sound card and ALSA setup). Then start *Sequencer64* so that it uses JACK for MIDI. Provide a default MIDI file so that all elements of the user interface can come into play. Also use the "&" character so that we get back to the command-line prompt. Finally, on our system the main synthesizer (*Yoshimi*) comes up on MIDI buss 5, so we add an option to remap all events to that buss:

```
$ seq64 --jack-midi --bus 5 contrib/midi/b4uacuse-seq24.midi &
```

If the `--alsa` option is used instead of `--jack-midi`, then the "Native" button will show "ALSA" instead.

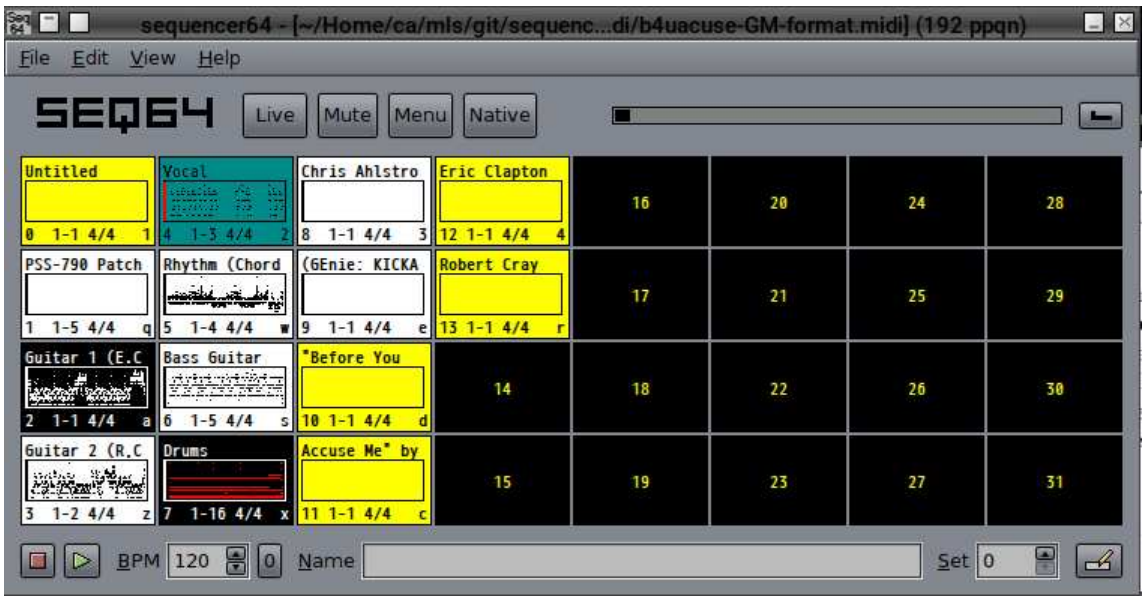


Figure 1: Sequencer64 Main Screen, Native JACK MIDI

Then the *Sequencer64* main window appears, as shown in figure 1 "Sequencer64 Main Screen, Native JACK MIDI" on page 12. It has some differences from the *Seq24* main window: the highlighting of empty patterns in yellow, a different font, additional control buttons, and much more that is not shown. Most of these items can be configured via the "rc" and "user" configuration files or command-line options. Note that the buttons show labels in the figure: "Live" or "Song", "Mute", "Menu", and "JACK", "ALSA", or "Native" (native JACK MIDI. Another new item that *is* shown is the button with a "0" as its label. This is the new **Tap Tempo** button (see section 3.3 "Patterns / Bottom Panel" on page 54for more information).

Tip: As with most user-interfaces, holding the mouse over some user-interface elements for a short period will let one view a description (tooltip) of what it does.

2 Menu

The *Sequencer64* menu, as seen at the top of figure 1 "Sequencer64 Main Screen, Native JACK MIDI" on page 12, is fairly simple, but it is important to understand the structure of the menu entries.

2.1 Menu / File

The **File** menu is used to save and load standard MIDI files. *Sequencer64* will handle any Format 0 or Format 1 standard files that other sequencers export. The *Sequencer64* menu entry contains the sub-items shown in figure 2 "Sequencer64 File Menu Items" on page 13. The next few sub-sections discuss the sub-items in the *File* sub-menu.

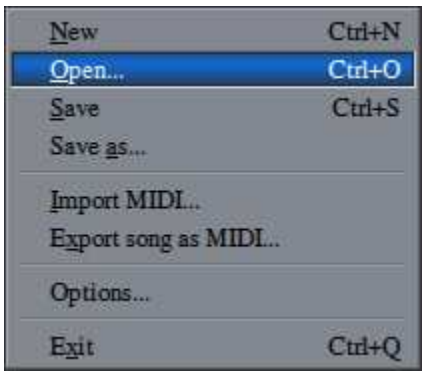


Figure 2: Sequencer64 File Menu Items

1. **New**
2. **Open...**
3. **Save**
4. **Save As...**
5. **Import MIDI...**
6. **Export song as MIDI...**
7. **Options...**
8. **Exit**

2.2 Menu / File / New

The **New** menu entry clears out any current song and patterns, allowing one to create new ones from scratch. If unsaved changes are pending, the user will be prompted to save the changes. Prompting for changes is more comprehensive than *Seq24*.

2.2.1 Menu / File / Open

The **Open** menu entry opens a song (MIDI file) that had been saved previously. It opens up a standard GTK+2 file dialog:

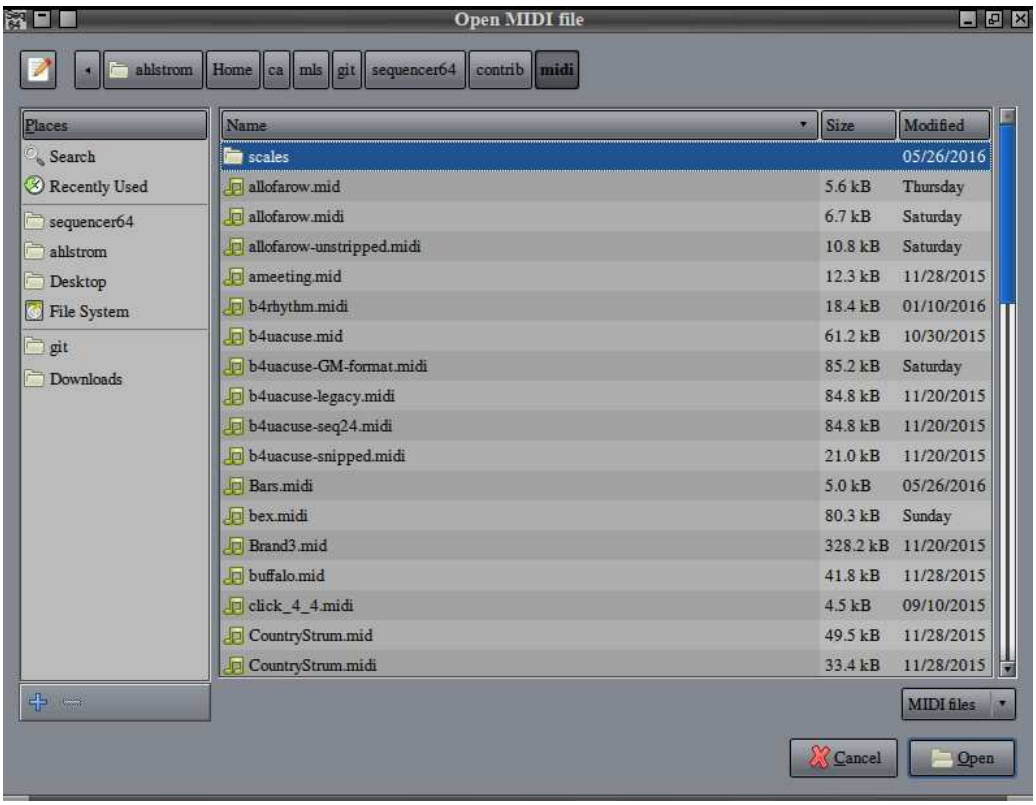


Figure 3: File / Open

This dialog lets one type a file-name, highlighting the first file (if any) that matches the characters typed so far.

If unsaved changes are pending, the user will (usually) be prompted to save the changes. When in doubt, save! If still in doubt, keep backups of your tunes!

2.2.2 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, then 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.

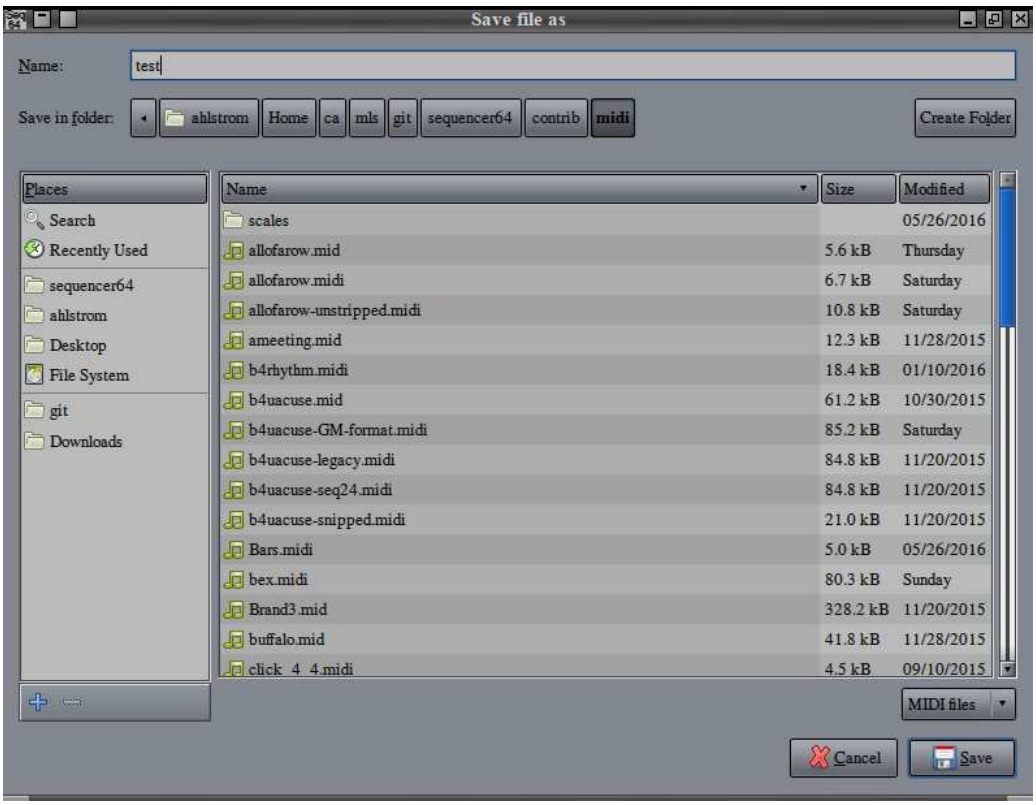


Figure 4: File / Save As

To save a new file, or to save the current existing file to a new name, enter the name in the name field, without an extension. *Sequencer64* 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:

```
myfile.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 *Sequencer64*, one sees that all of the labelling, pattern information, and song layout has been preserved in this file. Even the pattern layout (arrangement), as discussed in section 5.2.2 "Song Editor / Arrangement Panel / Piano Roll" on page 82, have been saved. (But the L and R marker positions are not saved.)

Compare the sizes of the original project MIDI file and the output MIDI file after *Sequencer64* saves them are different. The reason is that, after the last track in the file, a number of sequencer-specific (SeqSpec) items are saved, to preserve this extra information. In legacy mode, *Sequencer64* saves this information in the same format as *Seq24*. Otherwise, it saves it in an even more MIDI-compatible format. There is also an option to strip this extra information if it is empty.

New: In its normal mode, *Sequencer64* saves this information by marking each SeqSpec section as vendor-specific information, and marking this section as a regular MIDI track. The legacy and new formats of the final "track" are explained in section 13.2 "Legacy Proprietary Track Format" on page 144.

2.2.3 Menu / File / Import MIDI

The **Import** menu entry allows one to import an SMF 0 or SMF 1 MIDI file into one or more patterns, one pattern per track in the MIDI file. Even long tracks, that aren't short loops, are read in properly.

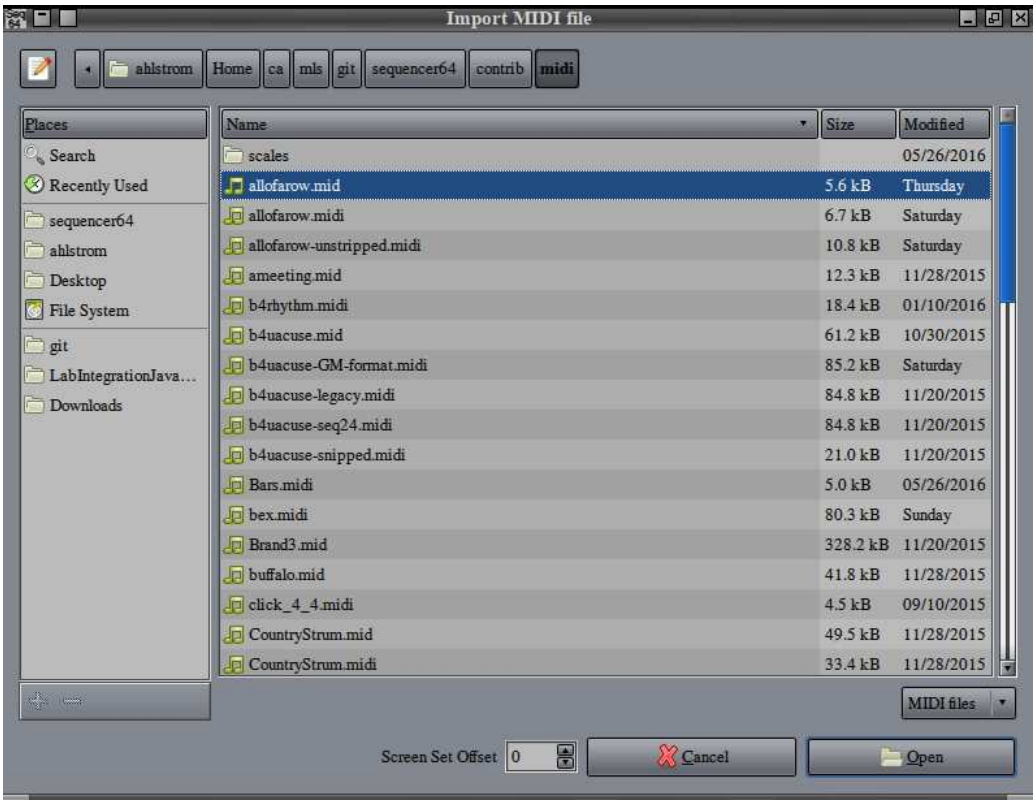


Figure 5: File / Import MIDI

When imported, each track, whether a music track or an information track, is entered into its own loop/pattern box. The import operation can handle reasonably complex files, such as the `contrib/b4uacuse.mid` file, which contains a transcription of an Eric Clapton / Robert Cray tune made over 20 years ago.

Note the additional file-dialog field, **Select Screen Offset**. This setting lets one place the imported data into a screen-set other than the first screen-set (screen-set 0). This field is not editable. It requires using the scroll button to move the screen set offset up or down in value. The legal values range from 0 to 31.

When the file is imported, the sequence number for each track read in is adjusted to put the track in the desired screen-set. The import can placed the imported data into any of the 32 available screen-sets. Quite large songs can be built up by importing patterns from other MIDI files.

The import operation also handles SMF 0 MIDI files. It parcels out the SMF 0 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?

2.2.4 Menu / File / Export song as MIDI

Thanks to the *Seq32* project, the ability to export songs to MIDI format has been added. "But wait,", you say, "Sequencer64 already saves to a MIDI-compatible format. Why the need for an Export function?" Well, the **Export song as MIDI** function modifies the song in the following ways:

- Only tracks (sequences, loops, or patterns) that are "exportable" are written. To be exportable, a track must be unmuted, and it must have triggers (see section 11.1.20 "Concepts / Terms / trigger" on page 133). That is, the track must have a layout entry in the **Song Editor**. A track need not have any playable data to be exported.
- All of the triggers for a given track are consolidated. Each trigger is used to generate the events, including repeats and offsets play of the track information. If there is a gap in the layout (e.g. due to the **Expand** operation in the Song Editor), then there is a corresponding gap in the events that are played. The result is one consolidated trigger that reconstructs the original playback layout. This trigger is not needed (or even usable) in any non-Seq24-based MIDI sequencer; the events themselves are sufficient to play the performance exactly. But the trigger is useful for further editing of the song/performance, as will be seen later, so it is kept (in a **SeqSpec** section).
- All the tracks that are exported are consolidated, so that any empty pattern slots between them are removed. No matter what set the original track was in, it ends up in the first set.
- Other additions, such as time signature and tempo meta events, are written in the same manner as for a normal **File / Save** operation.

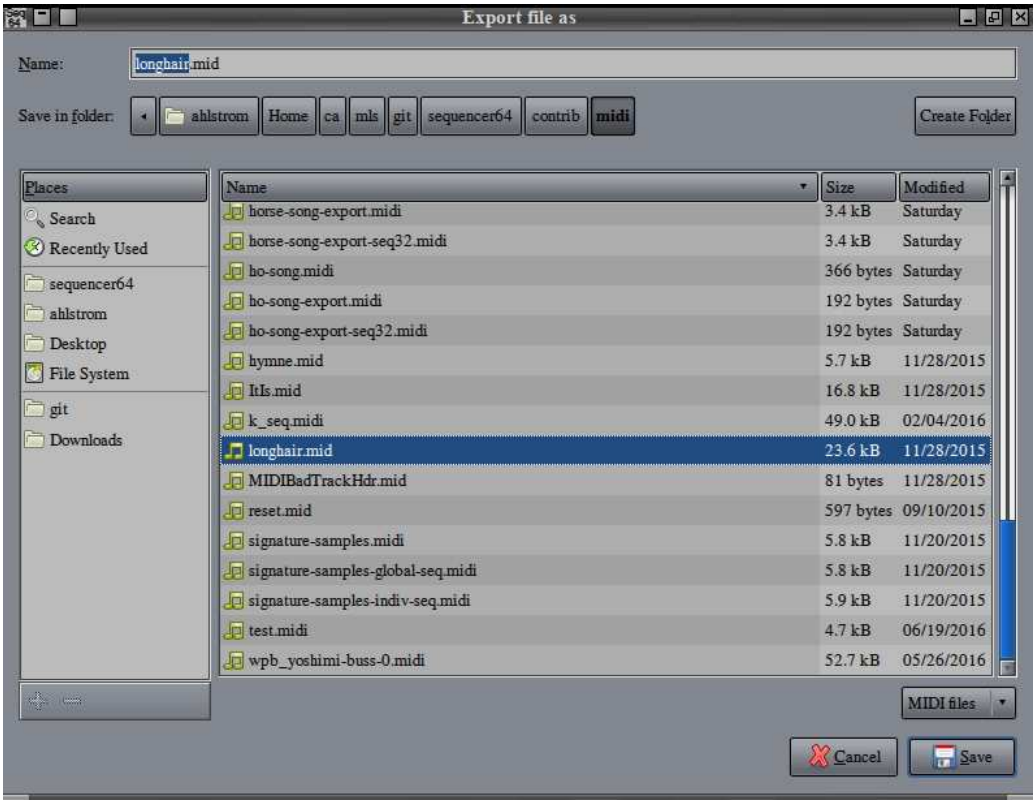


Figure 6: File / Export Song as MIDI

If there are no exportable tracks, the following message is shown:

Figure 7: Unexportable MIDI File

Once the file has been exported, it can be opened in order to see the results of the export. Both the main window view and the song performance view will show the results of the export.

Here is a short song, in two sets, shown in a composite view of four windows, showing each set and the performance layout of the tracks in the sets.

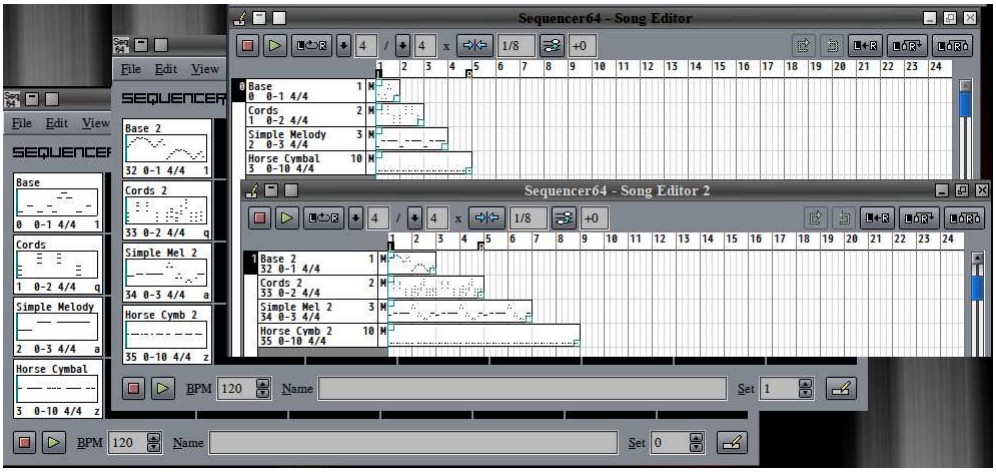


Figure 8: Composite View of Exportable Song

The left column shows the four tracks of the first set (Set 0), the next column shows the four tracks of the second set (Set 1), and the layouts of the two sets are shown in the remainder of the diagram.

Now let's export this song, and see the result:

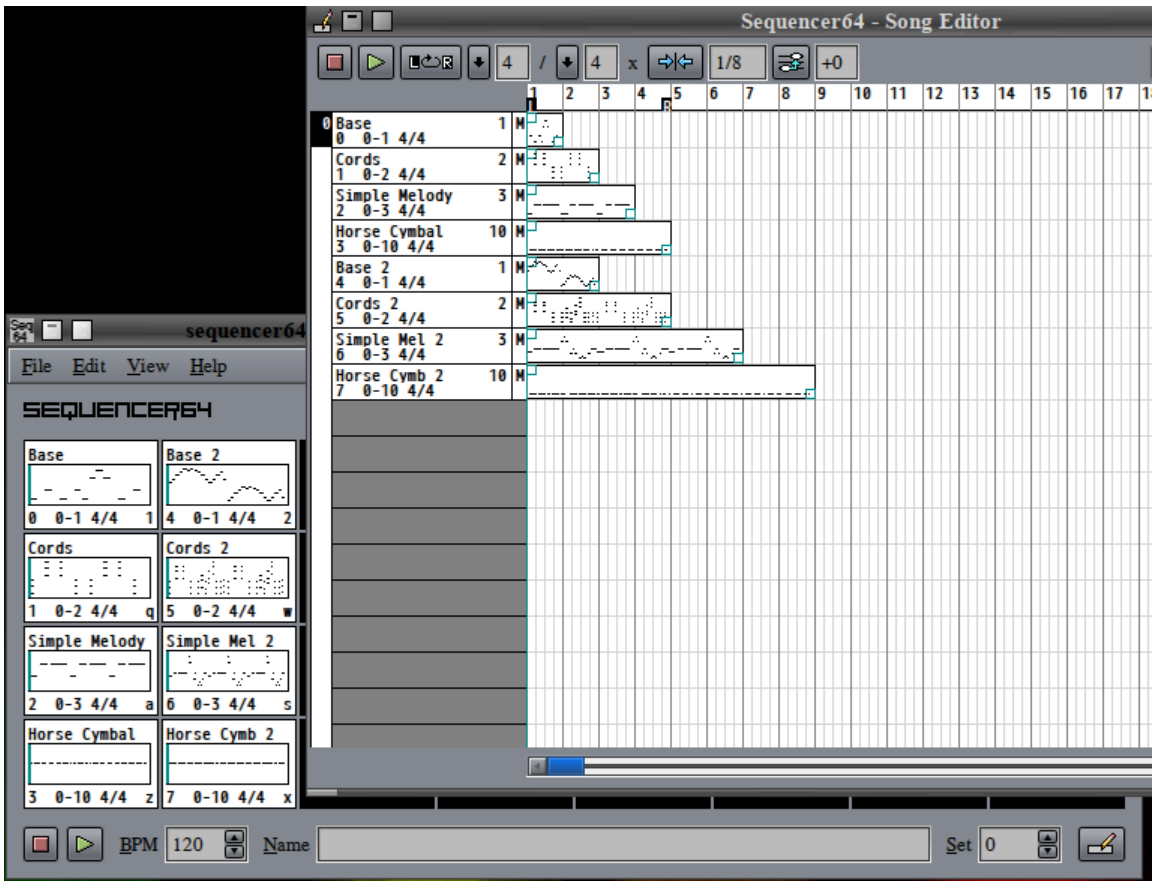


Figure 9: Composite View of Exported Song

Once can see that the two sets are now combined into the first set, and all of the track layouts (triggers) have been exported. Had there been gaps in layouts or repeats of layouts in the song/performance data, these would have been reflected in the triggers. Much more complex examples are, of course, possible.

2.2.5 Menu / File / Options

The **Options** menu item provides a number of settings in one tabbed dialog, shown in the figures that follow. This dialog allows one to select which sequence gets the MIDI clock, which incoming MIDI events control the sequencer, what keys are mapped to functions, how the mouse works, and some JACK parameters.

Note that there is now an extra tab-page for **Ext Keys**. It is not shown in some of the screenshots; we are too lazy to re-do those screenshot just to show the new tab.

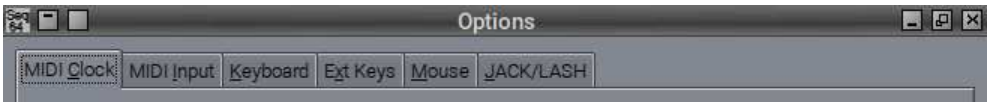


Figure 10: File / Options Tabs for Version 0.9.18+

2.2.5.1 Menu / File / Options / MIDI Clock

The **MIDI Clock** tab provides a way to send the MIDI clock to one or more of the *Sequencer64* output busses. It is used to configure to what busses the MIDI clock gets dumped. It also shows the devices that can play music. The items that appear in this tab depend on four things:

- What MIDI devices are connected to the computer. For example, MIDI controllers, USB MIDI cables, and other devices will add MIDI output devices (ports) to the system.
- What MIDI software devices are running on the computer. For example, running MIDI software synthesizers such as *Timidity* and *Yoshimi* will add extra output devices (playback ports) to a system.
- The setting of the "manual ALSA ports" option, `--manual-alsa-ports` command-line option or the `[manual-alsa-ports]` section of the `sequencer64.rc` configuration file, as described in section 8.8 "Sequencer64 "rc" File / Other Sections" on page 110
- The setting of the *Sequencer64*-specific "reveal ALSA ports" option, `--reveal-alsa-ports` command-line option or the `[reveal-alsa-ports]` section of the `sequencer64.rc` configuration file, as described in section 8.8 "Sequencer64 "rc" File / Other Sections" on page 110

For the current discussion, a USB MIDI cable was plugged into the system, and the *Timidity* and *Yoshimi* (in ALSA mode) software synthesizers were running. *Sequencer64* was also running, of course, with the option of "manual ALSA ports" (`-m` or `--manual-alsa-ports`) and ALSA (`-A` or `--alsa` turned on. Here are the devices shown by the ALSA MIDI playback command-line application:

```
$ aplaymidi -l
 Port    Client name                      Port name
 14:0    Midi Through                     Midi Through Port-0
 24:0    E-MU XMidi1X1 Tab                E-MU XMidi1X1 Tab MIDI 1
128:0    TiMidity                         TiMidity port 0
128:1    TiMidity                         TiMidity port 1
128:2    TiMidity                         TiMidity port 2
128:3    TiMidity                         TiMidity port 3
129:16   sequencer64                      sequencer64 in
```

Turning to figure 11 "MIDI Clock, Manual Option On" on page 21, note the 16 devices provided by *Sequencer64*. Also note that its first value is 1, not 0, due to the MIDI Thru port occupying slot 0. This figure shows the result with the "manual ALSA option" turned on. But remember that, now this option also applies to the native JACK MIDI version of *Sequencer64*, which has an executable named `seq64`.

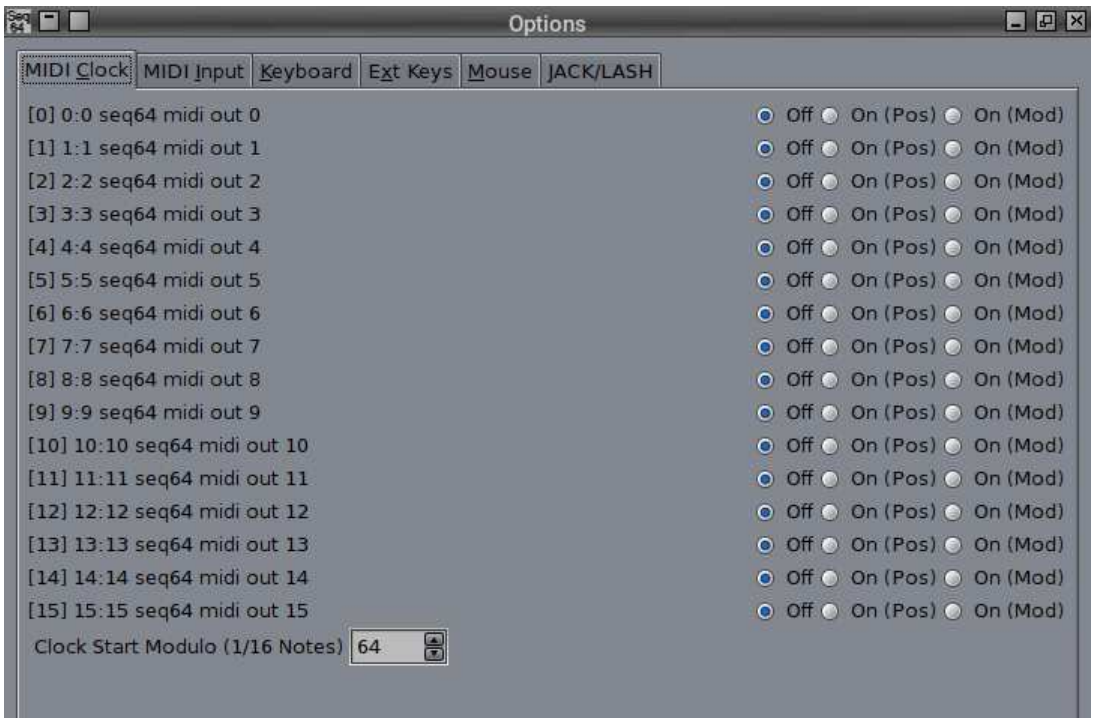


Figure 11: MIDI Clock, Manual Option On

It basically shows the 16 MIDI output busses that *Sequencer64* can drive. One would have to use a JACK or ALSA MIDI connection application to put 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 "user" configuration file can change the names of the ports, causing further confusion. The following elements are present in this dialog:

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

1. Index Number. The number in square brackets is simply an ordinal indicating the position of the output buss in the list of busses. It can be used with the `--bus` option to redirect all output to that one buss.

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 just matches the index number.

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

4. Buss Name. These labels indicate the output busses (ports) of *Sequencer64*. They range from **[1] sequencer64 1** to **[16] sequencer64 16** in the legacy application, `sequencer64`. They range from **[1] seq64 1** to **[16] seq64 16**. in the native JACK application, `seq64`.

5. Off. This setting disables the MIDI clock for the given output buss. However, note that MIDI output can still be sent to those ports, and each port that has a device connected to it will play music.

For feeding *Yoshimi* (running in ALSA mode) with MIDI data, we found that this setting is the one that must be made in order for *Yoshimi* to produce a sound.

6. On (Pos). The MIDI clock will be sent to this buss. MIDI Song Position and MIDI Continue will be sent if playback is starting at greater than tick 0 in Song mode. Otherwise, MIDI Start will be sent.

7. On (Mod). The 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.

8. Clock Start Modulo. Clock Start Modulo (1/16 Notes). This value starts at 1 and ranges on upward to 16384. It defaults to 64. It is used by the **On (Mod)** setting discussed above. It is the `[midi-clock-mod-ticks]` option in the *Sequencer64* "rc" file as described in section 8.8 "Sequencer64 "rc" File / Other Sections" on page 110.

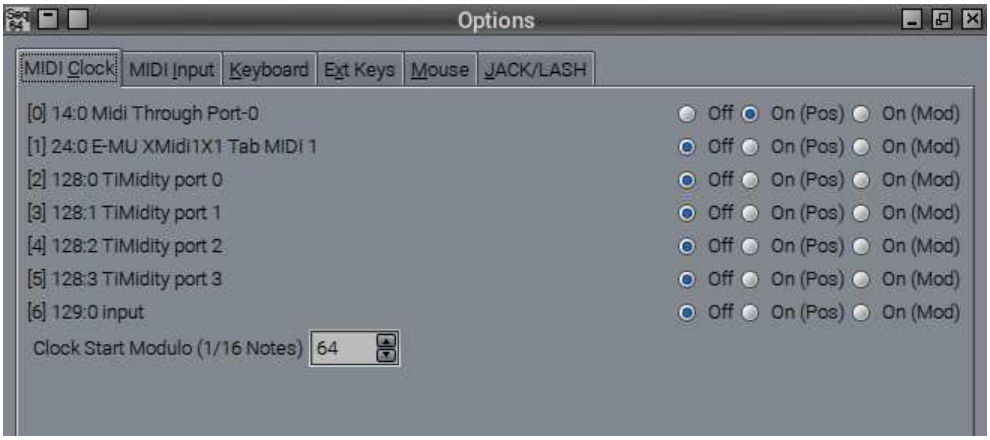


Figure 12: MIDI Clock, Manual Option Off (ALSA View)

As shown by the figure above, with the "manual ALSA option" turned off, all of the devices that can be driven by MIDI output are shown, including the MIDI Thru port, the MIDI port on the *E-MU XMidi1x1* USB cable, the four ports provided by *Timidity*, and the unlabelled port provided by the *Yoshimi* synthesizer running in ALSA mode. (However, `seq64` does show the name "yoshimi" as the client name.) One could theoretically play music through 6 or 7 devices using *Sequencer64* with this setup.

See section 15.2 "Sequencer64 Native JACK MIDI" on page 157for a lot more information about native JACK support, and examples of JACK MIDI ports and connections.

TODO: There is currently no user-interface item corresponding to the "manual ALSA" command-line and "rc" configuration file option. We also want to rename this option to simply "manual" in the near future.

2.2.5.2 Menu / File / Options / MIDI Input

To allow *Sequencer64* to record MIDI from MIDI devices such as controllers and keyboards, the output of the ALSA MIDI recording command-line application is relevant:

```
$ arecordmidi -l
 Port    Client name                      Port name
 14:0    Midi Through                     Midi Through Port-0
 24:0    USB2.0-MIDI                      USB2.0-MIDI MIDI 1
129:0    sequencer64                      [1] sequencer64 1
129:1    sequencer64                      [2] sequencer64 2
129:2    sequencer64                      [3] sequencer64 3
 . . .   . . .                              . . .
129:15   sequencer64                      [16] sequencer64 16
```

We see that we can record MIDI from the MIDI Thru port, from the USB MIDI cable, and MIDI from any of the 16 output ports provided by the manual ALSA port mode of *Sequencer64*.

If the "manual ALSA ports" option is turned on, then the only item in the **MIDI Input** tab is the single MIDI input buss provided by *Sequencer64*: **[0] sequencer64 0**, or, since the MIDI Thru port takes slot 0, **[1] sequencer64 1**.

Figure 13: MIDI Input, Manual Ports On (Condensed View)

This item, if checked, allows *Sequencer64* to be used to record MIDI information from another source (which must be connected to this port by another application), or pass it through to the output busses that are configured to allow pass-through (in the Pattern Editor, as discussed in section 4.4 "Pattern Editor / Bottom Panel" on page 72.)

Warning: If the `[user-midi-bus-definitions]` value in the "user" 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-alsa-ports` (`-r`) option.)

If the "manual ALSA ports" option is turned off, then the input ports from the system are shown:

Figure 14: MIDI Input, Manual ALSA Ports Off (Condensed View)

For example, one could check input #1 to have *Sequencer64* record MIDI from an old-fashioned MIDI keyboard that is connected to another USB MIDI cable (the *E-MU Xmidi*). If the keyboard didn't have a sound generator, one would also want *Sequencer64* to pass this MIDI on to a sound generator, such as a software or hardware synthesizer attached to one of the ports shown in figure 12 "MIDI Clock, Manual Option Off (ALSA View)" on page 22.

Warning: Again, note that the "user" configuration file can override what is actually displayed as hardware. If you define these sections, they should match your hardware exactly, and your hardware should not change from session to session.

Note the two sections of this configuration page:

Input Buses delineates the MIDI input devices as noted above. **Input Options** adds further refinements to MIDI input. The current option present

Record input into sequence according to channel, is a new (and *untested*) feature ported from *Seq32*. When enabled, MIDI input with multiple channels is distributed to each sequence according to what MIDI channel the sequence is set to. When disabled, the legacy recording behavior dumps all data into the current sequence, regardless of channel.

2.2.5.3 Menu / File / Options / Keyboard

Seq24 allows extensive use of keyboard shortcuts to make operations go faster than when using a mouse, and *Sequencer64* extends that tradition. The **Keyboard** tab allows for the configuration of these keyboard shortcuts.

Warning: There are a number of "gotchas" to be aware of when assigning keys to the fields in the **Keyboard** tab:

- Whenever one of the text fields in this dialog has the focus (and that is usually the case), then any keystroke, including keys like Ctrl, Alt, and Super (Mod4 or Windows key), can alter the value of a field to that of the keystroke. This change is very easy to do accidentally! Use the mouse to move this window and to click its **OK** button!
- Some of the keys traditionally used (or used by default) for control have been adapted for other uses. One example is `Ctrl-L`, which brings up the learn mode that can be started using the "L" button.
- *Sequencer64* has appropriated the Shift key so that it modifies a click on a pattern so that all of the other patterns are *toggled*. Therefore, using characters that require the Shift key while clicking,

such as { and }, when used to set the **Replace** function, becomes surprising. Instead, look to the remaining keys: `F11`, `F12`, and the "keypad" keys if more options are wanted. Be sure to look at the **Ext Keys** tab to see what other keys are in use.

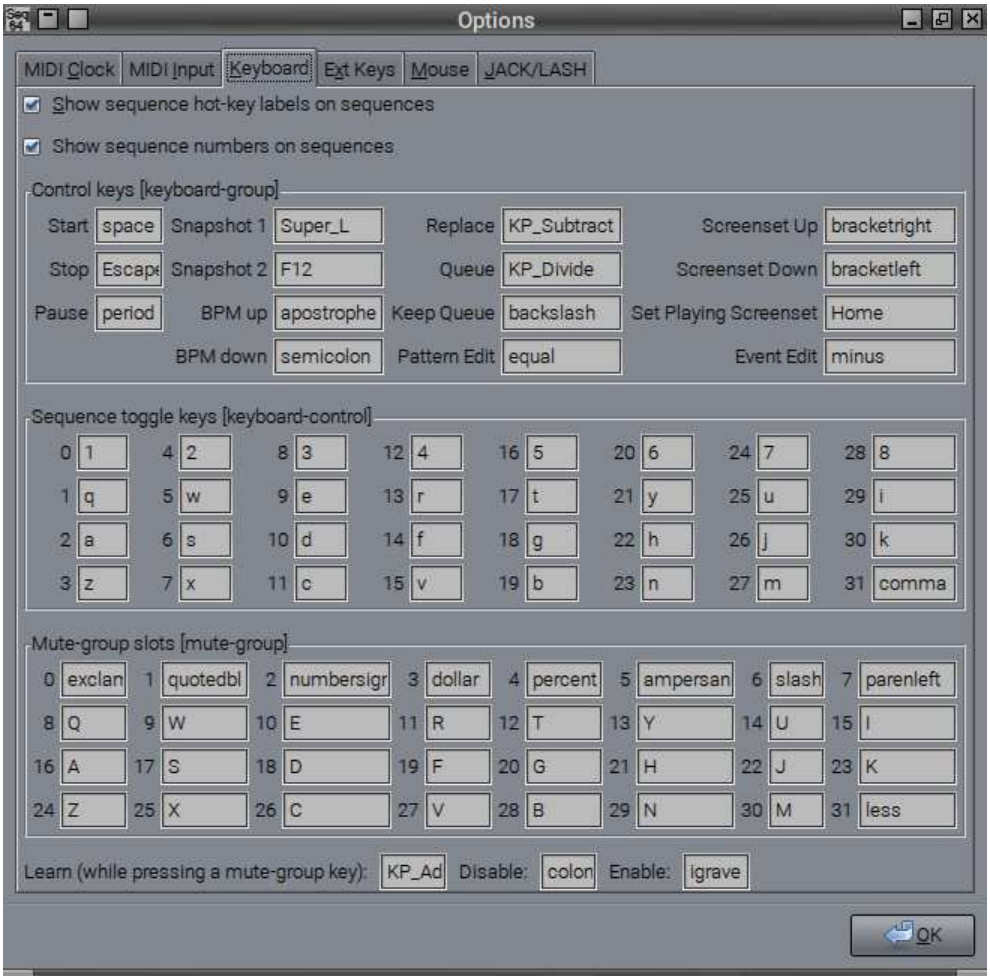


Figure 15: File / Options / Keyboard (Condensed View)

Note the new **Ext Keys** tab is not shown in this figure. We won't attempt to cover every user-interface item in this busy dialog, just the categories. Some items are discussed in other parts of this manual.

New: Also, if the application has been built with the pause option, an additional key definition is shown for the Pause key.

By default, the Pause key is the period (".") . An old version of the "rc" file is automatically fixed to include this new option. (The pause feature can be removed by rebuilding the application after configuring with the `--disable-pause` option.)

New: With version 0.9.12, we add a new feature as we try to achieve the goal of being able to edit a pattern using only the keyboard. *Sequencer64* now 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. These keys are currently hard-wired, but we'll make them configurable eventually.

To continue with a listing of the keyboard options:

1. **Show sequence hot-key labels on sequences**
2. **Show sequence numbers on sequences**
3. **Control keys [keyboard-group]**
4. **Sequence toggle keys [keyboard-control]**
5. **Mute-group slots [mute-group]**
6. **Learn**
7. **Disable**
8. **Enable**

1. Show key labels on sequence. This item, if enabled, shows the key labels in the lower-right corner of each loop/pattern in the Patterns window (the main window). This feature is useful for live playback and control of a song. Note that this option is also available in the "rc" configuration file.

2. Show sequence numbers on sequence. **New:** If this option is turned on, the empty slots in the Patterns window show the prospective sequence number. See the following figure for one look of this feature.

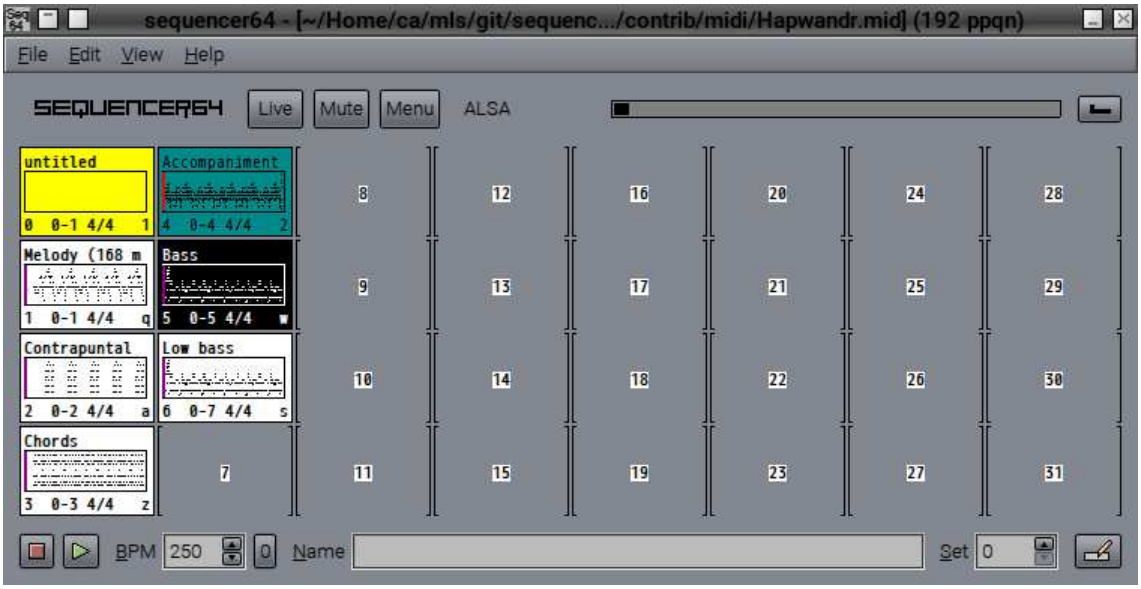


Figure 16: Pattern Window with Numbering

If one doesn't like it, turn off the option in the "rc" configuration file, or try other grid options in the "user" configuration file. Also note that the option also changes the visibility of sequence numbers in active sequences and in the Song Editor's names column.

3. Control keys. This block of fields in the **Options / Keyboard** tab provides shortcut keys for many operations of *Sequencer64*.

1. **Start**. Key: **space**.
2. **Stop**. Key: **Escape**.
3. **Pause**. (new) Key: **period**.
4. **Snapshot 1**. Key: **Alt_L**.
5. **Snapshot 2**. Key: **Alt_R**.
6. **bpm up**. Key: **apostrophe**.

7. **bpm down**. Key: **semicolon**.
8. **Replace**. Key: **Control_L**.
9. **Queue**. Key: **Control_R**.
10. **Keep queue**. Key: **backslash**.
11. **Screenset down**. Key: **bracketleft**.
12. **Screenset up**. Key: **bracketright**.
13. **Set playing screenset**. Key: **Home**.

Note that 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). Also note that the keys definable in this tab are only a subset of the various keys that can be used, especially keys used with the `Ctrl` key or other modifier keys.

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.

To *queue* a pattern means to ready it for playback upon 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 starts. A pattern can be queued by holding the queue key (defined in **File / Options / Keyboard / queue**) and pressing a pattern-slot shortcut key. Instead of the pattern turning on immediately, it turns on at the next repeat of the pattern.

The *keep queue* functionality allows the queue to be held without holding down the queue button the whole time. First, press the keep-queue key (defined in **File / Options / Keyboard / Keep queue**). Now, hitting any of the shortcut keys, no matter how many, sets up the corresponding pattern slot to be queued. This mode is disabled by hitting the "queue" key (any currently active queues remain active until finished).

4. 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, and can also be shown in each loop/pattern box. No need to list them all here! Please note that we often call them "shortcut keys" where the context makes it clear that they apply to the armed/unarmed state of a pattern. Sometimes they are called "hot keys".

5. Mute-group slots. Each of these keys operates on the mute-grouping of one of the 32 loop/pattern boxes. These keys are layed out logically on the keyboard, and can also be shown in each loop/pattern box. No need to list them all here!

One thing we need to explain is just what mute-grouping means functionally.

Mute groups are shortcuts to play a defined group of patterns on the current set, while stopping other patterns from the current set, and all patters from other sets.

To define the group of patterns for one mute group, press and hold the Learn key (the `Insert` key by default, but see the current configuration) and, simultaneously, press one of the mute group keys: *Sequencer64* will save the currently-playing pattern slots into the corresponding mute group. Note that the default mute group keys need the `Shift` modifier, but one does not need the `Shift` key while pressing `Insert` to learn the group, only to trigger it (*Sequencer64* will automatically assign the corresponding key with `Shift` activated).

Group-mute can be globally enabled or disabled (with default keys apostrophe ' and igrave or grave `). So make sure it is enabled before trying to use it.

6. Learn. Learn (while pressing a mute-group key). This items sets the key used to initiate a learn mode. It is the **Insert** key by default. 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*. It is new to *Sequencer64*.

7. Disable. It is the **apostrophe** key by default. This key is the *group off* key.

8. Enable. It is the **igrave** (back-tick) key by default. This key is the *group on* key.

2.2.5.4 Menu / File / Options / Ext Keys

A number of additional functions have been added to *Sequencer64*, and keystrokes have been provided for those new functions, in the **Ext Keys** page.

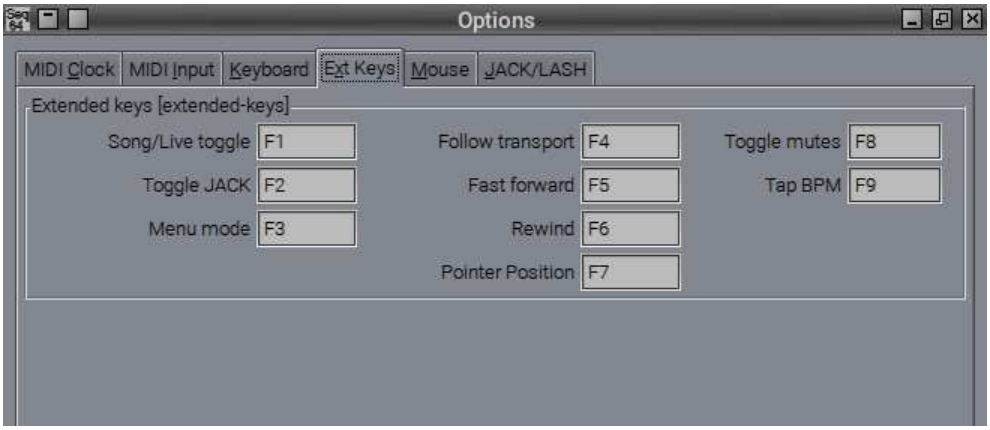


Figure 17: File / Options / Ext Keys (Condensed View)

9. Ext Keys. This block of fields in the **Options / Ext Keys** tab provides shortcut keys for more operations of *Sequencer64*, many of them ported from *Seq32*. The default keys are shown.

1. **Song/Live toggle**. Key: **F1**.
2. **Toggle JACK**. Key: **F2**.
3. **Menu mode**. Key: **F3**.
4. **Follow transport**. Key: **F4**.
5. **Fast forward**. Key: **F5**.
6. **Rewind**. Key: **F6**.
7. **Pointer Position**. Key: **F7**.
8. **Toggle mutes**. Key: **F8**.
9. **Tap BPM**. Key: **F9**.

Most of these extended keys implement operations performed with button presses. However, some of the new keystrokes do not have a corresponding button (yet). Also, we still need to clarify the meaning of some of the new functions.

These values are saved as the `[extended-keys]` section of the "rc" configuration file. However, not all builds of *Sequencer64* will support the new keys. Some of the new features can be enabled or disabled during the build-configure step, and one's favorite distro may decide to disable some features. In this case, although the values will still be stored in the "rc" file, they will be disabled in this tab:

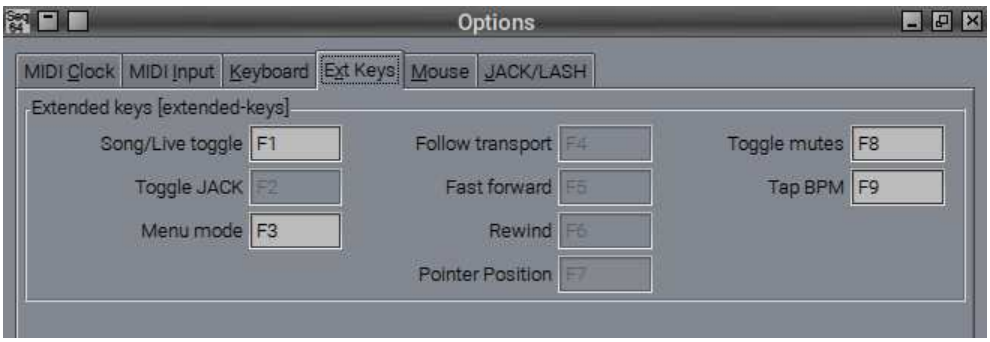


Figure 18: File / Options / Ext Keys (Disabled)

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 hotkey value, which defaults to `F1`.

The *JACK mode* is normally set via the **File / Options / JACK / JACK Connect** or **JACK Disconnect** buttons. But, if *Sequencer64* is built for using the *Seq32* JACK support, then this keystroke will toggle between JACK connect and JACK disconnect. Note that, with this kind of build, the **Song Editor** will also have a **JACK** button. The hotkey for this function defaults to `F2`.

The *menu mode* simply indicates if the main menu of the main window is accessible or not. It will be disabled in playback so that more hotkeys can be used without triggering menu function. It can also be disabled by the user; the default hotkey is `F3`. Here is Stazed's explanation of the feature, mildly edited:

> *"why disabling is needed when playing"* The original seq24 had numerous conflicts between the menu key binding and the default seq24 key binding for the mainwind sequence triggers. For example: Ctrl-q (quits the program without prompt). If you place a sequence in the default 'q' slot, you cannot use it with Ctrl-l or Ctrl-r (default replace or queue) because the menu grabs the keys. Same goes for the Alt-l or Alt-r (default snapshot 1 or 2). Try same as above with Alt-f, Alt-v, Alt-h, Ctrl-n, Ctrl-o... etc. So I just shut off all the menus by default when playing because it seems that they should not be needed then... especially in a live performance.
>
> *"why a button?"* On occasion I wanted to use the mainwnd key binding when stopped to set the sequences to be ready before starting. It's also a sort of safety feature as well, just toggle the menus off before going live so that you don't hit Ctrl-q, Ctrl-n etc. forgetting things are not playing....

The *follow transport* functionality is a feature ported from *Seq32*. We'll have more of a description of it later. The default key is `F4`.

The *fast forward* functionality is a feature ported from *Seq32*. Basically, while this key is held, the song pointer will fast-forward through the song. The default key is `F5`. This feature does not (yet) have a corresponding button. This feature also requires that the *Seq32* transport option be enabled at build time.

The *rewind* functionality is a feature ported from *Seq32*. The default key is `F6`. Basically, while this key is held, the song pointer will rewind in the song. This feature does not (yet) have a corresponding button. This feature also requires that the *Seq32* transport option be enabled at build time.

Note that, since these keys are use only in the Song Editor, the *seq32* keys `f` and `r` keys could be used as well. Be sure not to use the following keys, which are already hardwired for other functions in the Song Editor:

- `p`. Paint mode.

- `x`. Escape paint mode.

The *pointer position* functionality is a feature ported from *Seq32*. The default key is `F7`. Basically, 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. This feature also requires that the *Seq32* transport option be enabled at build time, which is the case if the *Seq32* JACK build option is selected (and now it is the default).

The *toggle mutes* function 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 hotkey value is stored in the "rc" configuration file, and defaults to `F8`.

The *tap bpm* function 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 button for this function. After 5 seconds, this feature resets automatically, so the user can try again if not satisfied. At least two clicks are needed for this functionality to work.

2.2.5.5 Menu / File / Options / Mouse

This item selects the mouse-interaction method.

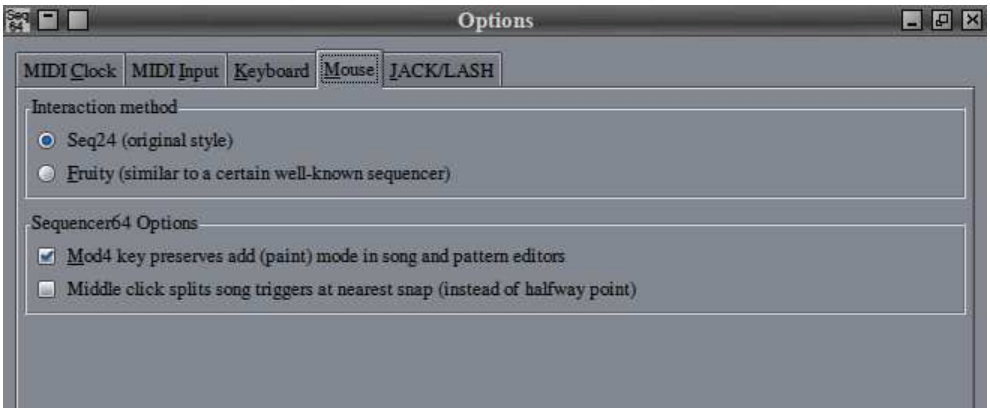


Figure 19: File / Options / Mouse (Condensed View)

Interaction Method

The default mouse interaction method is **Seq24 (original style)**. The alternate mouse interaction method is **Fruity (similar to a certain well known sequencer)**.

The alternate method is presumably that of the *Fruity Loops* (now *FL Studio*) sequencer. The fruity mode seems to involve the following (based on scanning the source code):

- **Left-click left side**. Begin a grow/shrink operation for the left side.
- **Left-click right side**. Begin a grow/shrink operation for the right side.
- **Left-click middle**. Move the object.
- **Left-click**. Add an event if nothing selected.
- **Middle-click**. Split the note?

The *Seq24* "original style" is pretty much 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 the one must first *click and hold* the right mouse button, and then *click and drag* the right mouse button to insert notes, Notes are inserted to be at the current length and grid-snap values for the sequence editor for as long as the left button is pressed. Notes are inserted only up to the boundary of the sequence length. And, once notes are inserted, moving the mouse with the left button still held down simply 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 strange, but it is a powerful way to layer notes into a short sequence. We call it the "draw mode" or "paint mode".

Note that 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.

Sequencer64 Options

This section currently contains a couple of new options.

Mod4 key preserves add (paint) mode in song and pattern editors

In order to work better with certain trackpads, the "Seq24" mode of mouse interaction can be modified in the Pattern or Song editors so that the Mod4 key (Super or Windows key) can be pressed when releasing the right mouse button. This keeps the mouse in note-add mode. Another right-click, without pressing Mod4, will exit this mode. The reason for this feature is the crummy FocalTech touchpad on one of the author's laptops. This trackpad seems to have only a single button, which the driver interprets as left or right depending where the finger is when it is clicked. There's no way to click the right and left buttons at the same time. There's no way to make a middle-click action. What a crock!

Note that this option will not interfere with the Mod4 key being set in the **Keyboard** option tab, since the keys there mainly apply to the Patterns Panel (main window).

Middle click splits song triggers at nearest snap (instead of halfway point)

Another way to turn on the paint mode has been added, based on a feature found in a patch that someone posted about in some mailing list somewhere on the internet. To turn on the paint mode, press the `p` key while in the sequence editor. This is just like pressing the right mouse button, but the draw/paint mode sticks (as if the Mod4 mode were in force). To get out of the paint mode, press the `x` key while in the sequence editor. These keys, however, do not work (currently) while the sequence is playing.

These convenience options are limited to the pattern/sequence editor window and the performance editor window, and may need some heavier testing. Also note that some *Sequencer64* windows can use the ctrl-left-click as a middle click.

2.2.5.6 Menu / File / Options / Jack Sync and LASH

This tab sets up options for JACK synchronization, if *Sequencer64* was built with JACK support. (Why wouldn't it be?) It now also supports native JACK MIDI. This tab also sets up options for using LASH session management, if *Sequencer64* was build with LASH support.

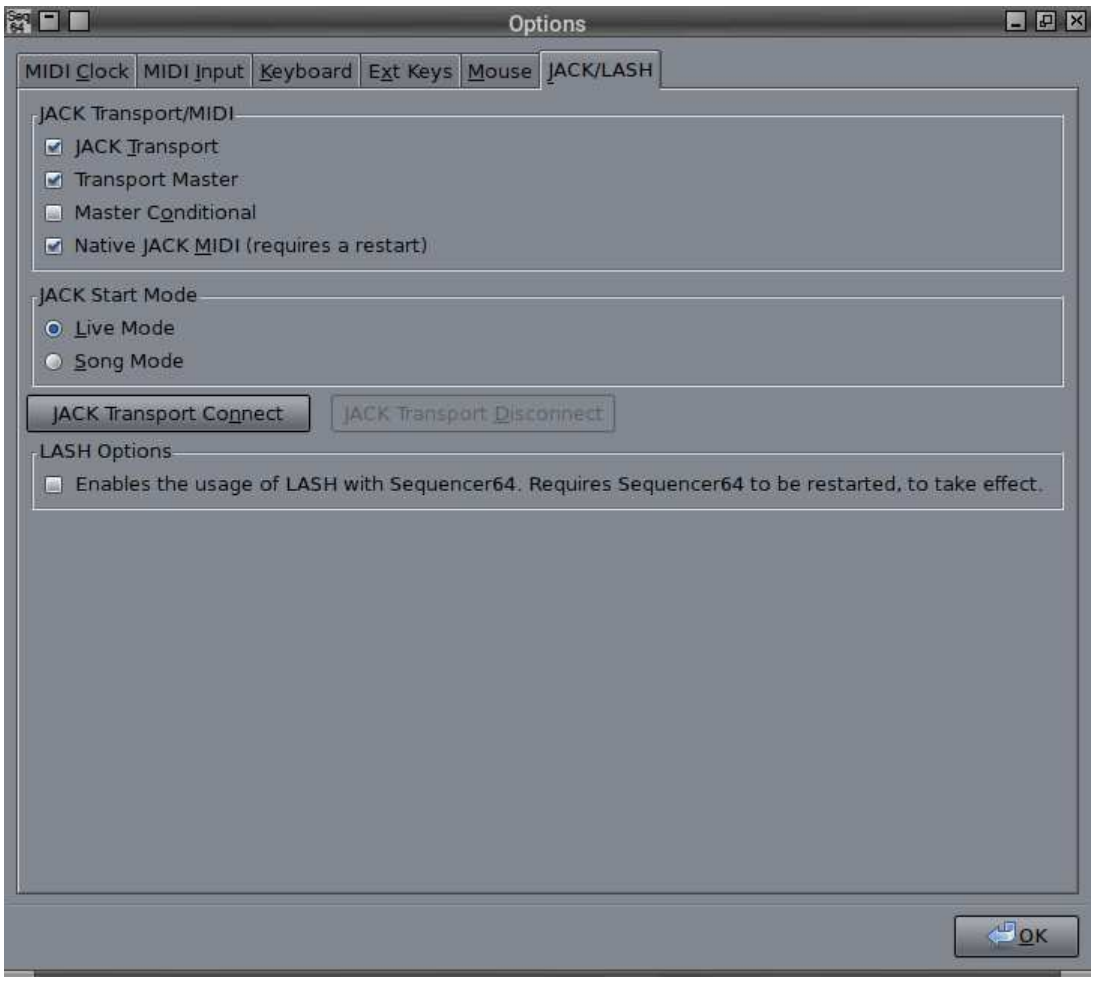


Figure 20: File / Options / JACK/LASH

The main sections in this dialog are:

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

1. Transport/MIDI. These settings are stored in the "rc" file settings group `[jack-transport]`. See section 8.7 "Sequencer64 "rc" File / JACK Transport" on page 109, which describes this configuration option. 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, *Sequencer64* can start and stop playback.
- **Transport Master**. *Sequencer64* will attempt to serve as the JACK Master. The command-line option is `--jack-master`. **Tip**: *Sequencer64* generally works a little better as JACK master.
- **Master Conditional**. *Sequencer64* will fail to serve as the JACK Master if there is already a Master set. The command-line option is `--jack-master-cond`.
- **Native JACK MIDI**. This option is for the `seq64` version of *Sequencer64*. If set, MIDI input and output occur using the new JACK MIDI interface, rather than using ALSA. However, if JACK is

not running on the system, then `seq64` will fall back to ALSA mode. The command-line option is `--jack-midi`.

Note that there are long-standing issues with the JACK support of *Seq24*, and *Sequencer64* currently inherits some of them, in spite of some bug fixes. Generally, if one experiences issues in transport control, try making one of the other sequencer applications the JACK Master. If one starts *Sequencer64* in JACK mode without JACK running, it will take a little while for *Sequencer64* to start up, and it will fall back to ALSA usage.

Finally, 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 *Sequencer64*... the settings are automatically saved when *Sequencer64* 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-playback 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`.

Note that, in ALSA mode (non-JACK mode), *Sequencer64* now *does* select the playback modes according to which window started the playback. We have reverted back to legacy *Seq24* behavior.

The main window, or pattern window, causes playback to be in live mode. The user can arm and mute patterns in that windows, 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". Of course, in JACK mode, it selects them according to the chosen live/song mode as discussed above.

3. Connect. Connect to JACK Sync. This button is useful to restart JACK sync when making changes to it, or when *Sequencer64* was started in ALSA mode.

4. Disconnect. Disconnect from JACK Sync. This button is useful to stop JACK sync when making changes to it.

5. LASH Options. Currently contains only one item, which enables the usage of LASH session management. Currently, *Sequencer64* needs to be restarted to complete the enabling or disabling of LASH support. Like the rest of the options, this one is written to the "rc" configuration file.

Finally, there is a new button (labelled *Master* in the following figure) in the main window to bring up directly the **JACK** (or **JACK/LASH**) page.

Figure 21: JACK Connection Button

This button not only brings up the JACK page, but also shows the current status of the MIDI connection: **Master** (JACK Master), **JACK** (JACK Slave), **Native** (native JACK MIDI), and **ALSA**. Currently, the `Ctrl-P` will also bring up this page.

2.3 Menu / Edit

The **Edit** menu has undergone some expansion lately.

Figure 22: Edit Menu

1. **Song Editor...**
2. **Apply song transpose**
3. **Clear mute groups**
4. **Reload mute groups**
5. **Mute all tracks**
6. **Unute all tracks**
7. **Toggle mute all tracks**

6. Song Editor. This item is the same as the **View / Song Editor toggle** menu entry. It toggles the presence of the main song editor.

7. Apply song transpose.

Selecting this item applies the song transposition value to **all** sequences/patterns that are marked as transposable. (Normally, drum tracks are *not* transposable). This actively changes the note/pitch value of all note and aftertouch events in the pattern.

Once the transpositions are done, the transposition value is set to 0. For the setting of song transpose, see section 5 "Song Editor" on page 76, for more information. Also note that transpose can be enabled in the patterns panel for each pattern (see section 3.2.2 "Pattern" on page 45) and in the sequence editor (see section 4 "Pattern Editor" on page 55).

8. Clear mute groups.

A feature of *Seq24* and *Sequencer64* is that the mute groups are saved in both the "rc" file (see section 8.3 "Sequencer64 "rc" File / Mute-Group Section" on page 105) and in the "MIDI" file (see section 13.2 "Legacy Proprietary Track Format" on page 144).

This menu entry clears them. If the MIDI file is saved, it then has no mute-group information. And then, currently, if the application is exited, the clear mute-group information is also saved. We'd like to be able to handle the "rc" and "MIDI" mute-groups separately in the future.

9. 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.

10. Mute all tracks.

This menu entry, available only in **Live** mode, immediately mutes *all* patterns in the entire song.

11. Unmute all tracks.

This menu entry, available only in **Live** mode, immediately unmutes *all* patterns in the entire song.

12. Toggle mute all tracks.

This option toggles the mute/armed status of **all** tracks. It is only available in **Live** mode, which overrides **Song** mode even if the Song Editor is focussed.

*Do not confuse it with the main **Mute** button, which toggles the status only of the tracks that are armed and remembers them.*

2.4 Menu / View

If the "allow two perfedits" option is turned off in the "user" configuration file, this menu item has only one entry, **Song Editor**, which is already covered by a button at the bottom of the Patterns window. Selecting this item bring up the Song Editor window. See figure 67 "Song Editor Window" on page 77. The Song Editor window can also be brought up via the Ctrl-E key.

If the **allow two perfedits** option is turned on in the "user" configuration file, this menu item has two entries, as shown in the following figure:

Figure 23: Dual Song Editor Entries in View Menu

Note that only the first Song Editor has a user-interface button and a hot-key. Also note that there can be issues bringing up the second song-editor with the hot-key. The menu entry will always work.

If two song editors are up, they each track any changes made in the other song editor. But the main purpose of two song editors is to arrange two different parts of the performance at the same time.

2.5 Menu / Help / About...

This menu entry shows the "About" dialog.

Figure 24: Help / About

That dialog provides access to the credits for the program, including the authors and the project documentors. It has recently been updated to show Git version-control information as well.

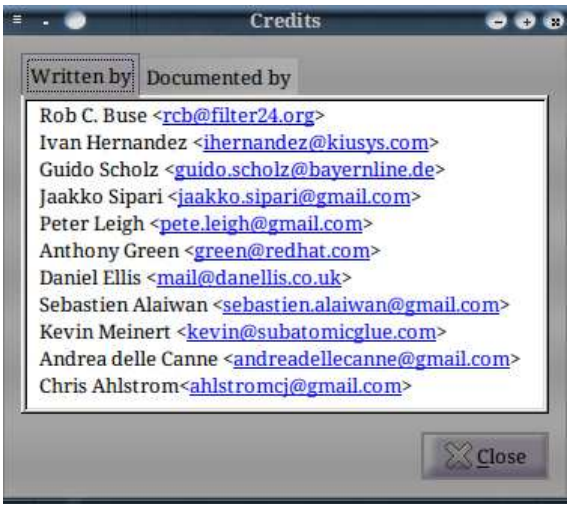


Figure 25: Help Credits

Shows who has worked on the program, with the original author at the top of the list.

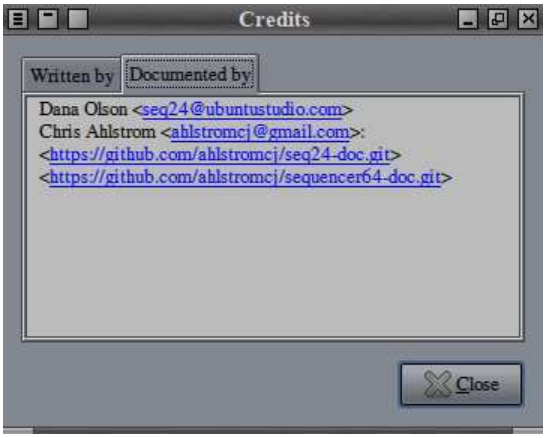


Figure 26: Help Documentation

Shows who has documented this project.

2.6 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 the one of the following command lines:

```
$ sequencer64 --version
$ seq64 --version
```

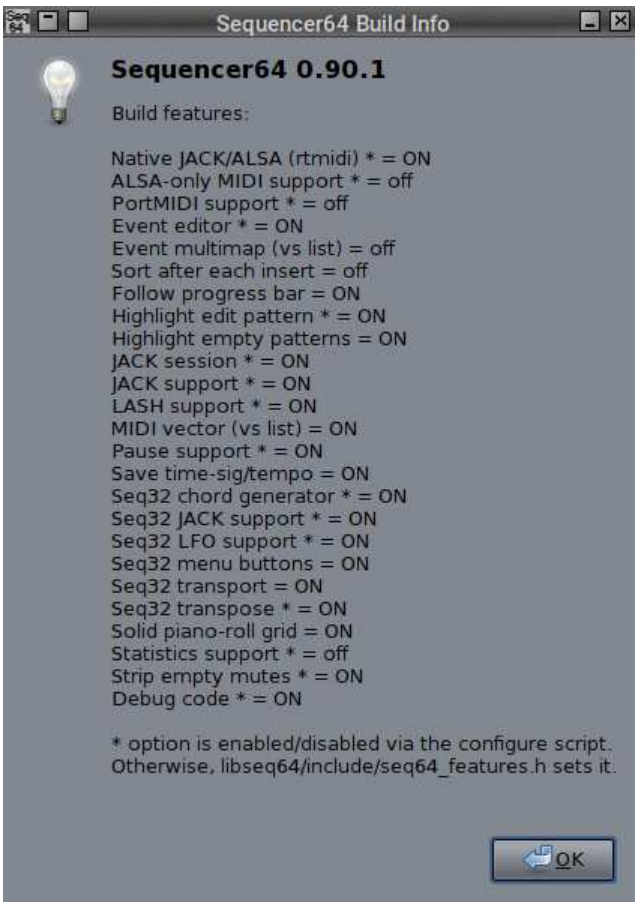


Figure 27: Help / Build Info

3 Patterns Panel

Sequencer64 works with patterns (loops, track, or sequences) that are repeated all along a song. One composes and edits small patterns, and combines them to create a full song. This is a powerful way to work, and makes one productive within an hour.

The *Sequencer64 Patterns Panel* is the main window of *Sequencer64*. See figure 1 "Sequencer64 Main Screen, Native JACK MIDI" on page 12. It is also called the "main window" or the "patterns window". It is here one creates a set of patterns (see section 11.1.16 "Concepts / Terms / screen set" on page 133), manages the configuration, and opens the pattern or song editors.

When the Patterns Panel has the application focus (in ALSA mode), and *Sequencer64* is *not* running in JACK mode, it puts *Sequencer64* 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.

If the song editor (see section 5 "Song Editor" on page 76) has the input focus in ALSA mode, then it controls the muting/unmuting of each pattern, and *Sequencer64* runs in "song mode". However, if *Sequencer64* is using JACK transport, then, instead of the behavior described above, live versus song mode is controlled by the JACK start mode option (see the **JACK Start mode** item in section 2.2.5.6 "Menu / File / Options / Jack Sync and LASH" on page 31).

Back to the patterns panel.... For exposition, we break the Patterns Panel into a menu bar, a top panel, a pattern panel, and a bottom panel. (The *Sequencer64* menu bar is discussed in section 2 "Menu" on page 12.)

3.1 Patterns / Top Panel

The top panel of the Pattern window is simple, consisting of the name of the program and a couple of controls. The original version looked like this:

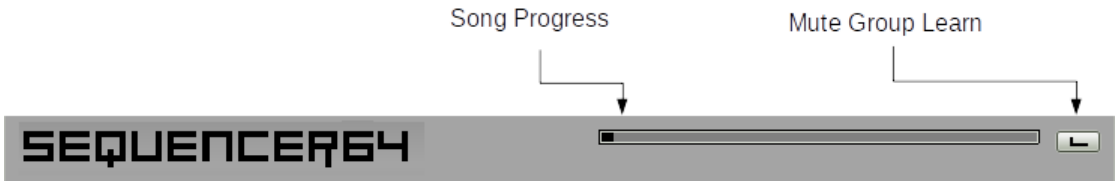


Figure 28: Patterns Panel, Top Panel, Older Version

But, if compiled with `SEQ64_STAZED_MENU_BUTTONS` defined, there are some extra buttons available:

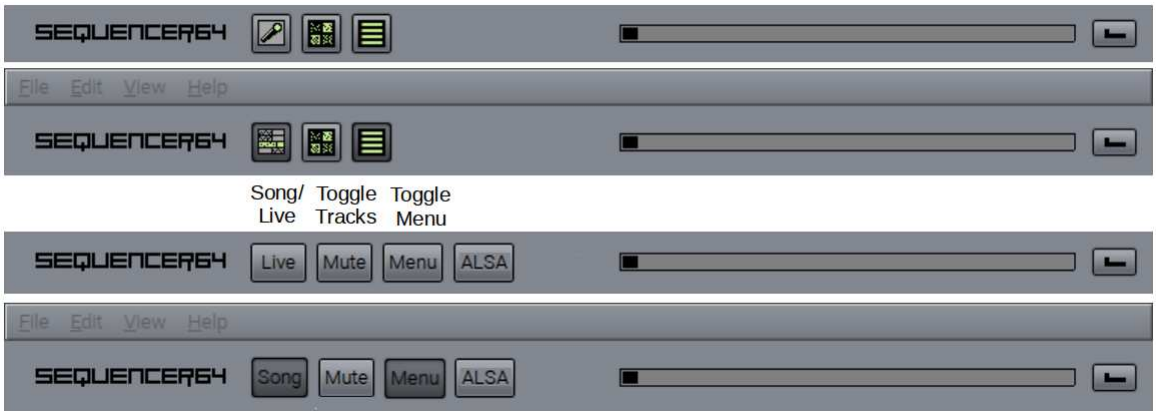


Figure 29: Patterns Panel, New Top Panel Items

This figure shows the possible appearance of the main window top panel. If *Sequencer64* was built with `SEQ64_MENU_BUTTON_PIXMAPS` defined, then the buttons have icons on them; otherwise they have text. The figure shows what they look like, and how they look with the song/live and toggle-menu functions activated.

1. **Song/Live**
2. **Toggle Playing Tracks**
3. **Toggle Menu**
4. **ALSA/JACK Mode**
5. **Song Progress**
6. **Mute Group Learn**

1. Song/Live. This new button allows the Song mode to be in force even if the **Song Editor** does not have the focus of the application. However, if the **Song Editor** does have focus, it overrides this button,

to preserve expected behavior. There is also a configurable hotkey associated with it, which defaults to `F1`, and is configurable in the "rc" file and the **File / Options / Ext Keys** page.

2. Toggle Tracks. This button changes the status of all of the playing tracks, reversing the mute status of each pattern that is playing. The next click will then unmute those tracks. Because it can be confusing, this button will be disabled (not shown in the figure) in Song mode. There is also a configurable hotkey associated with it, called **Toggle mutes**, which defaults to `F8`, and is configurable in the "rc" file and the **File / Options / Ext Keys** page.

3. Toggle Menu. This button enables and disables the main menu of the main window. Disabling it allows additional hotkeys to be used without calling forth menu entries. There is also a configurable hotkey associated with it, called **Menu mode**, which defaults to `F3`, and is configurable in the "rc" file and the **File / Options / Ext Keys** page.

4. ALSA/JACK Mode. This button used to be just a label, but now it can help set the ALSA versus JACK modes. When clicked, it brings up the JACK connection page from the **File / Options** dialog. The hot-key for this button is `Ctrl-P`.

5. Song Progress. The **Song Progress** bar is also known as the "main time" bar. This bar shows a number of small black cursors ("pills") that show the progress of the song through the various patterns. For short patterns, the progress is fast. For patterns that last longer, the progress is slow. The whole field flashes in time with the beat. This field shows that something is going on. It can also indicate the relative lengths of the various patterns.

Note that the individual pattern boxes in the main panel, for patterns that are not empty, have their own moving progress cursor, a vertical line in each box. By default, this progress line is black, but it can be changed to other colors via a "user" configuration file entry in the `[user-interface-settings]` section. Set the `progress_bar_colored` item to a value ranging from 0 (black) to 6 (dark cyan). There is also a `progress_bar_thick` item to enable a thicker progress bar, for better visibility.

6. Mute Group Learn. This button is also known as the "L" button. Click this button, and then press a mute-group key to store the mute-state (whether armed or unarmed) of all the patterns with in that key. In addition to clicking this button, one can also press the `Ctrl-L` combination (analogous to the `Ctrl-E` keystroke that brings up the Song Editor). 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*.

See the **File / Options / Keyboard** menu entry to bring up the dialog showing the available mute-group keys and the corresponding hot-key for the "L" button. This key is usually the `Insert` key. Unlike the button or the `Ctrl-L` key, this key must be held down while pressing the desired mute-group key. This is a clumsy thing on our main keyboard... one must press and hold `Shift-Fn-Insert` to get the `Insert` key, and then also press the desired mute-group key. A real knuckle-buster! We have our own alternate setup (stored in the file `sequencer64.rc.example`. And we've also added the "shift-lock" feature for this reason.

Group-learn is a modifier key to be pressed just before pressing the desired group key, and the group on/off keys are there to enable each group, *not* to toggle group states.

To set up the mute groups, press the 'L' button, and then press a key on the keyboard to 'learn' or 'save' the preset. Looking at the list of keys assigned for these mute groups (in **File / Options / Keyboard**), the first bank of keys are "!", "", "?", etc., and the second bank are "Q", "W" "E", etc. Note that these keys do not toggle patterns, but simply turn them on (they unmute them, i.e. arm them). If the key is valid, then the following prompt occurs:

Figure 30: Group Learn Confirmation Prompt

When you ask the program to 'learn' the key, one cannot use the Shift key, so one normally could not use the "!" or other symbol keys. In fact, the process stops as soon as the Shift key is pressed:

Figure 31: Group Learn Failure Prompt (Shift Key)

New: To avoid this issue with the Shift key, *Sequencer64* now "Shift-Locks" the keys for you, so that none of the keys, whether letters or the punctuation characters above the numbers, need the Shift key to be held. There is now no need to use the **Caps Lock** is *On* before starting the 'learn' process. We did this because we tend to map the `Caps Lock` key as the *true* Control key, so that `Caps Lock` is no longer available (it is one of the most useless keys *ever*).

Once that works, one can configure the MIDI settings in similar ways by assigning MIDI commands to arm or toggle loops, using the 'on' option in the "rc" file. See section 8.1 "Sequencer64 "rc" File / MIDI Control Section" on page 96.

One can set the playing status of up to 32 previously defined mute/unmute patterns (groups) in the active screenset, similar to hardware sequencers. One can mute-unmute (according to the group definition) all loops in the playing screenset, which is the only one that can have sequences playing if a mute group is selected (like a live sequencer).

This arming is done either by one of the *group arm* keys or by a MIDI controller, both assigned in the `/.config/sequencer64/sequencer64.rc` or `/.seq24rc` files, and easily modified in the **File / Options / Keyboard** tab. These characters can be shown in each pattern (and in 0.9.11 and above, no matter which screenset is active).

A mute/unmute pattern (group) is stored by holding a *group learn* key (`Insert` by default) while pressing the corresponding *group arm* key. There are also keys assigned to turn on/off the group functionality. 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 (some set the `Home` key for this purpose). So, for

example, if one sets `A` to turn on the patterns in slots 1 and 2 in set 0 (the first set), then pressing `A` in another set will arm the patterns in the same relative location in that set. This setup is flexible, but takes some thought. One can set up a number of mute-groups, and decide to use them for all sets, or mentally allocate one mute-group per set. Please note that a mute-group key does not *toggle* the saved armed patterns... it can only turn them on.

Everything is configurable in the "rc" file.

3.2 Patterns / Main Panel

The main panel of the Patterns window provides a grid of empty boxes, each box delimited by brace-like lines at left and right. Each filled box represents a loop or pattern. One sees only 32 loops at a time in the main panel (but many more than 32 loops can be supported by *Sequencer64*). This group of 32 loops is called a "screen set", as discussed in section 11.1.16 "Concepts / Terms / screen set" on page 133. 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. There are a total of 32 sets, for a total of 1024 loops/patterns. Only one screen set can be controlled at a time, according to other notes we have found; have not yet tried to verify this assertion. But any number of screensets can be playing at the same time.

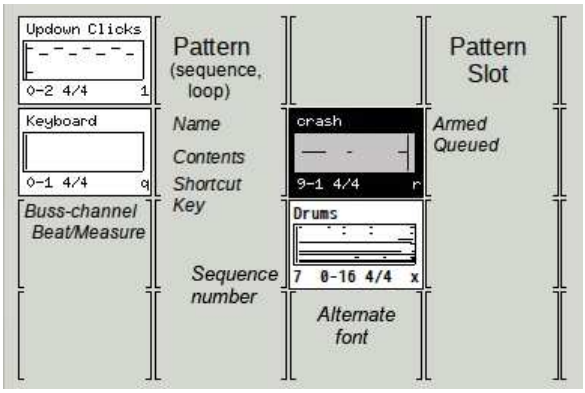


Figure 32: Patterns Panel, Main Panel Items

The individual items annoted in this figure are described in section 3.2.2 "Pattern" on page 45, in more detail. **However**, this figure does not show the new feature where the sequence number appears at the bottom left of the slot. 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**

3.2.1 Pattern Slot

An empty box is a slot for a pattern.

A pattern can show a number of different statuses based on the coloring of elements in the pattern slot.

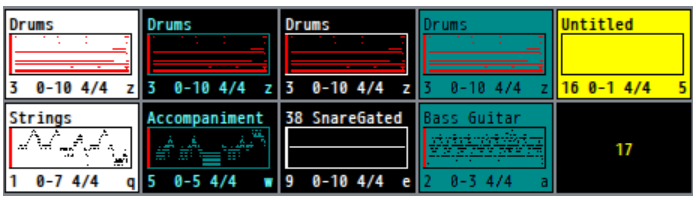


Figure 33: Various Status of Pattern Slots

The colors have meaning:

- **Empty background**. Whether the classic gray pattern of *Seq24*, or the many patterns of *Sequencer64*, including all black with yellow sequence numbers, this slot coloring indicates that the slot is unused.
- **White background**. Unarmed (muted) patterns show black text on a white background.
- **Black background**. Armed (unmuted) pattern. If the text is yellow, it is a pattern with no MIDI events, but is armed. Note that armed/unmuted patterns can be exported if they have a layout in the Song Editor.
- **Yellow background**. A pattern with no MIDI events, just textual MIDI information. If armed (uselessly), it is yellow text on a black background (not shown).
- **Cyan background, black text**. An unarmed pattern currently being edited in a pattern editor or event editor. Or, if an SMF 0 MIDI file was just opened or imported, this color combination indicates the SMF 0 format track with all of the data in the song, which only occurs in slot 16 (unless the user then dragged it to another slot).
- **Black background, cyan text**. An armed pattern currently being edited in a pattern editor or event editor. Or an armed SMF 0 format MIDI sequence.
- **Red events**. Indicates a pattern for which the new transpose feature is disabled. The white, black, and cyan background have the same meanings as in the other items for statuses of unarmed, armed, and currently being edited.

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

Figure 34: Empty Pattern, Right-Click Menu

1. **New**
2. **Paste**
3. **Song**

- Mute All Tracks
- Unmute All Tracks
- Toggle All Tracks
- Toggle Live Tracks

1. New. Creates a new loop or pattern. Clicking this menu entry fills in the empty box with an untitled pattern, and brings up the Pattern Editor so that one can fill in the new pattern.

In addition to right-clicking and selecting **New**, the user can double-click on the empty slot, to bring up a new instance of the sequence editor. For the double-click, the effect can be a bit confusing at first, because it currently also toggles the arming/mute status of the slot quickly twice (leaving it as it was), as well. It might take some getting used to, but we miss it when using *Seq24*.

A new feature is hitting the equals ("=") key, then hitting a pattern shortcut key (hotkey), to bring up a new sequence or edit an existing one in a pattern editor. Another new feature is hitting the minus ("-") key, then the hotkey, to bring up the event editor. The configuration file settings for the the '=' and '-' keys can be altering in the **File / Options / Keyboard** tab.

When an unarmed (muted) pattern is first brough up for sequence editing (or event editing), the slot in the main window is now highlighted, using black text on a cyan background, as being the "currently-edited" slot. (This is the same background used to indicate the original track in an SMF 0 to SMF 1 conversion.)

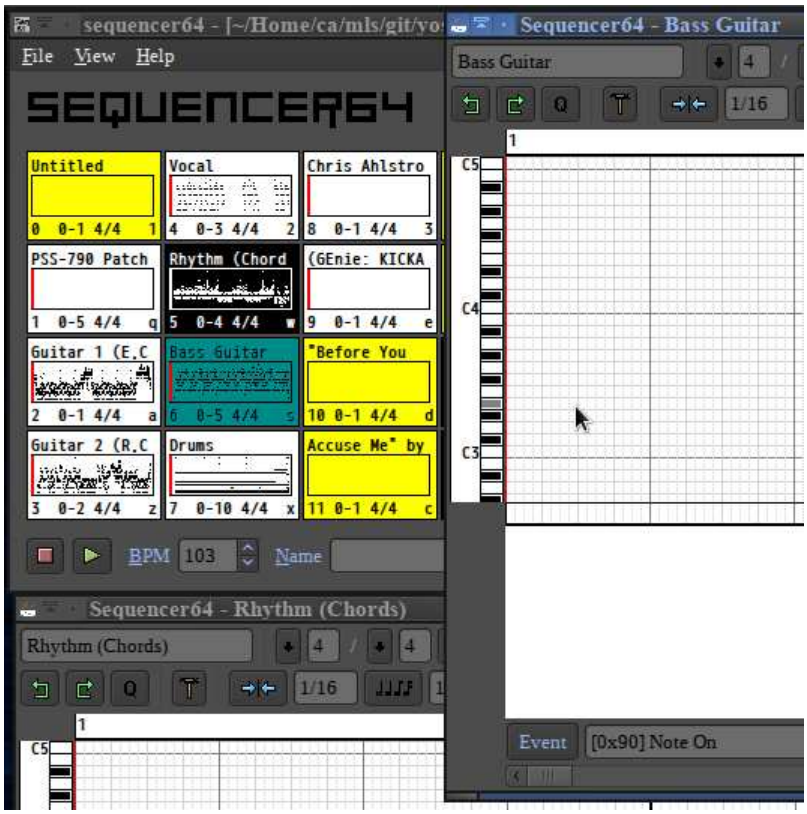


Figure 35: Currently-Edited Pattern, Unarmed

If the currently-edited sequence is armed (unmuted), then the highlighting is reversed (cyan text on a black background), and resembles the highlighting for an armed sequence (which is white text on a black background).

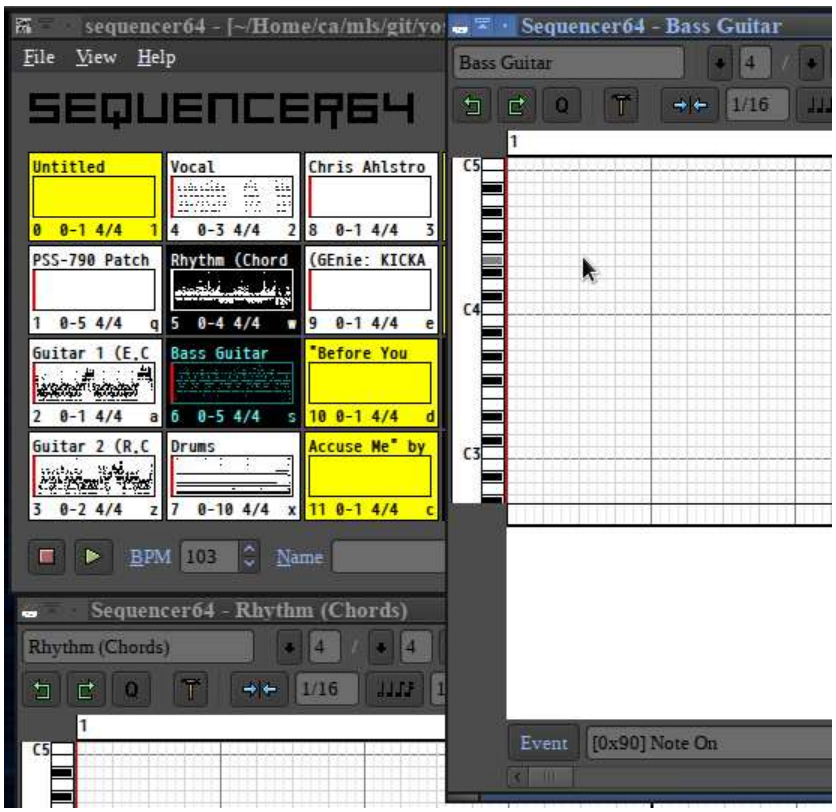


Figure 36: Currently-Edited Pattern, Armed

If more than one sequence or event editor is brought up, only the slot for the last one to have focus is hightlighted. Note that this highlighting also applies to the (new) Event Editor.

2. Paste. Pastes a loop or pattern that was previously copied. Also note that there is no `Ctrl-V` key for this operation in the main window.

3. Song. The **Song** items are described later, in reference to figure 39 "Existing Pattern, Right-Click Menu, Song" on page 48.

3.2.2 Pattern

A filled pattern slot is referred to informally as a *pattern* (or *loop*, or *sequence*). A pattern is shown in the Pattern windows as a filled box with the following items of information in it. Examine figure 32 "Patterns Panel, Main Panel Items" on page 42; it shows these items annotated for clarity.

- **Name**. This line contains the name or title of the pattern, to help reference it when juggling a number of patters.
- **Contents**. The contents of the pattern provide a fairly detailed and distinguishable representation of the notes or events in the pattern. Also, when the song is playing, a vertical bar cursor tracks the position of the playback of the pattern or loop; it returns to the beginning of the box every time that pattern starts over again. **New:** With *Sequencer64*, an imported empty pattern will no longer needlessly scroll. However, if a pattern has even a single event (say, a program change), it will scroll. **TODO:** It might be good to have some patterns marked as one-shot patterns. They

play once at the start of playback, and that is it. They could be marked with a cyan background. Currently, it is easy enough to use the Song Editor for this purpose, but then one cannot play the patterns in live mode.

- **Sequence Number**. If the option to show the sequencer number is set in the **File / Options / Keyboard** section (see section 2.2.5.3 "Menu / File / Options / Keyboard" on page 24, the this number is shown at the bottom left of the pattern slot.
- **Bus-Channel**. This pair of numbers shows the the MIDI buss number, a dash, and the MIDI channel number. For example, "0-2" means MIDI buss 0, channel 2.
- **Beat**. 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**. If the display of shortcut keys is enabled (see section 2.2.5.3 "Menu / File / Options / Keyboard" on page 24), then the key noted in the lower-right corner of the pattern can be pressed to toggle the mute/unmute status of that pattern. This action is an alternative to left-clicking on the pattern.
- **Progress Cursor**. At the left of each box is a vertical line, waiting for playback to start so that it can move through the pattern, again and again.
- **Armed**. See figure 32 "Patterns Panel, Main Panel Items" on page 42; it shows a black and white pattern. The black color 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 left-clicking on it. If the Shift key is held while left-clicking on a pattern, then the armed/unarmed state of every other active pattern is toggled. This feature is useful for isolating a single track or pattern.
- **Queued**. That same pattern also shows that it is queued, which means that it will toggle its playing status when the pattern next begins again.
- **Alternate font**. Later builds of *Sequencer64* are now built with a new font. See figure 32 "Patterns Panel, Main Panel Items" on page 42. It shows the new font. The old font can be selected in the "user" configuration file, and is also selected automatically if *Sequencer64* is run in the *legacy* mode.
- **Sequence number**. Later builds of *Sequencer64* are now built with the option to also show the sequence number in the pattern box, if the "show sequence numbers" option is on. This option can be set in the "user" configuration file. See figure 32 "Patterns Panel, Main Panel Items" on page 42. It shows an example of the sequence number, using the new font.

Left-clicking 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. Note that the armed status can also be toggled using hot-keys.

Also note that, if the Song Editor is the active window and was used to start the playback, the pattern boxes will toggle between the muted/unmuted states as the music plays, and the pattern is active or inactive at the point of playback. (The Song Editor acts as a list of triggers).

By right-clicking on an already-filled box, one brings up a menu to allow one to edit a existing one, or perform a few other actions specified in the context menu. Here is that menu:

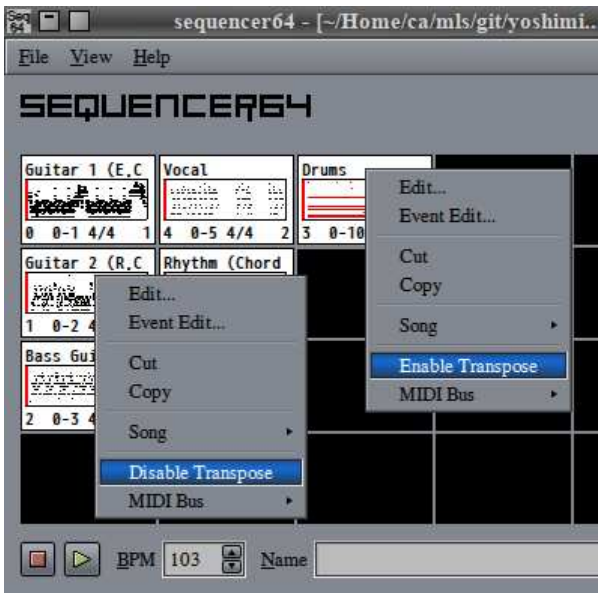


Figure 37: Existing Pattern, Right-Click Menus

Here one can choose to edit the pattern, cut and copy the pattern, set the MIDI bus/channel, and more. One can also clear all performance data for the pattern.

1. **Edit...**
2. **Event Edit...**
3. **Cut**
4. **Copy**
5. **Song**
6. **Enable or Disable Transpose**
7. **MIDI Bus**

1. Edit.... Edits an existing loop or pattern. Clicking this menu entry brings up the **Pattern Editor** so that one can modify the existing pattern by click-dragging new notes in a piano roll user-interface. See figure 47 "Pattern Edit Window" on page 56. Also known as the "sequence editor".

New: In addition to right-clicking and selecting **Edit...**, the user can double-click on the empty slot, to bring up the sequence editor.

New: Another way to bring up a pattern in the pattern editor is to click the **equal** key and then the pattern's shortcut key. For example, "**= q**" will open up the editor for the pattern with the hot-key **= q**. The Equals key (=) is the default key that does this action. This key can be changed by modifying the **File / Options / Keyboard / Control keys / Pattern Edit** item.

2. Event Edit.... **New:** Edits an existing loop or pattern, but using a detailed event editor that shows events as text and numbers, and allows editing them as text and numbers. Clicking this menu entry brings up the **Event Editor** so that one can view and modify the events in the existing pattern. See figure 47 "Pattern Edit Window" on page 56.

New: Another way to bring up a pattern in the event editor is to click the **minus** key and then the pattern's shortcut key. For example, "**- q**" will open up the event editor for that pattern. The Minus key (-) is the default key that does this action. This key can be changed by modifying the **File / Options / Keyboard / Control keys / Event Edit** item.

There are two things to note about the Event editor. First, this editor is not the same as the **event** pane in the pattern editor – the new event editor shows all events at once, and shows them only in text/list format. Second, this editor is very basic, meant for viewing MIDI events and making some minor edits or deletes on them. See section 6 "Event Editor" on page 84, for more information.

Now, in order to simplify the application, and avoid editing a pattern in two different dialogs, if either the pattern editor or the event editor is active for a given sequence, the right-click sequence-slot menu leaves out the **Edit...** and **Event Edit...** menu entries. This trimmed menu looks like this:

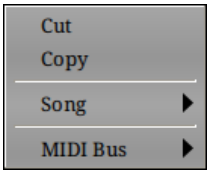


Figure 38: Existing Pattern, Right-Click Menu Without Edit Entries

The old functionality was to have the **Edit...** menu entry simply raise the existing pattern editor to the top of the windows.

3. Cut. Deletes and copies an existing loop or pattern. Note than one can also drag-and-drop a pattern into another cell. Also note that there is no `Ctrl-X` key for this operation in the main window.

4. Copy. Copies an existing loop or pattern. The pattern can then be pasted elsewhere in the Patterns panel. See section 3.2.1 "Pattern Slot" on page 42. Also note that there is no `Ctrl-C` key for this operation in the main window.

5. Song. Clicking this menu entry brings up a small popup menu:

Figure 39: Existing Pattern, Right-Click Menu, Song

1. **Clear This Track's Song Data**
2. **Mute All Tracks**
3. **Unmute All Tracks**
4. **Toggle All Tracks**
5. **Toggle Live Tracks**

Clear This Track's Song Data This item is not available if the pattern is empty. Selecting this filled-box right-click menu item causes that box's loop/pattern to be removed from the song editor. This means that it disappears from the Song Editor window, and so will not be played when the song plays in Song mode.

Song / Mute All Tracks Selecting this filled-box right-click menu item causes the tracks in the Song Editor to be muted. Sometime it takes a few seconds for the user-interfaces to show this big change. This item mutes all tracks (or loops/patterns). It works when one has opened the Song Editor window and started playing in playback mode by starting play using that window.

So, let us assume the song is running in playback mode. The patterns that are active (unmuted) in by that playback window are shown with a black background in the main patterns window. If one right

clicks on a pattern cell and selects **Song / Mute All Tracks**, all those patterns will become white and be silenced. Eventually, the Song Editor window catches up and shows the "M" activated for all tracks.

Unmute All Tracks Provides the opposite functionality, making all tracks armed and audible. Selecting this filled-box right-click menu item causes the tracks in the song to be unmuted.

Toggle All Track Toggles the armed/mute status of all tracks. It doesn't matter if Live or Song Mode is in force.

Note that there is also a feature where a `Shift-Left-Click` on a pattern slot toggles the mute status of all of the other *tracks*.

Toggle Live Tracks Toggles the mute status of only the armed/unmuted tracks when in Live mode. Works only in Live mode. This operation unmutes all tracks that are currently unmuted. The statuses of these armed tracks are saved; when this operation is performed again, those tracks are unmuted, turned back on. This menu entry provides the same function as the **Mute** button in the main window.

6. Disable or Enable Transpose. This menu entry changes depending upon whether the new transpose feature is enabled or disabled for the sequence/pattern. Note that, if the events shown in the slot are red, this denotes that transpose is currently *disabled* for that pattern.

7. MIDI Bus. Selecting this filled-box right-click menu item brings up a list of the 16 MIDI output busses that *Sequencer64* supports:

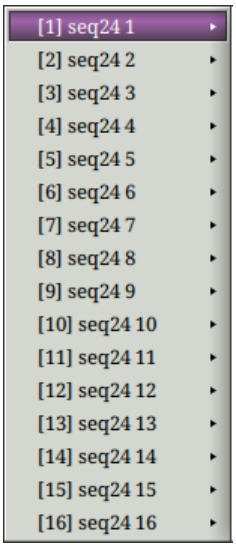


Figure 40: Existing Pattern, Right-Click Menu, MIDI Output Busses

New: Note that another way of specifying the busses is to supply the new `--buss n` option. This option is currently available only from the command line. It causes *every* pattern in the MIDI file to be allocated to that buss number when loaded, and when a new sequence/pattern is created. This option is meant for convenience or testing. If you save the file, it will then have that buss number as part of each track's data, which makes the song file just a little less portable.

For each of these buss items, another pop-up menu allows one to specify the MIDI output channel for that buss:

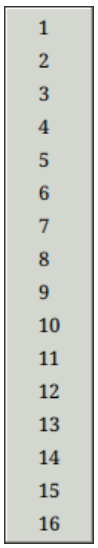


Figure 41: Existing Pattern, Right-Click Menu, MIDI Bus Ports

3.2.3 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.

3.2.3.1 Pattern Keys

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

For each pattern, hitting its assigned shortcut 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. This grid can be changed in the Keyboard options tab, and is saved in the *keyboard-control* section of the "rc" file.

```
[ 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. These hot-keys can be modified.

The `[` and `]` keys on the keyboard switch between sets, either decrementing or incrementing the set number.

The left and right `Alt` keys are, by default, set up in the **File / Options / Keyboard / Snapshot 1** and **Snapshot 2** fields to be used as "snapshot" keys. Our preference is to use something that does not trigger desktop commands, perhaps `F11` or `F12`, or one of the keys in the keypad.

When one of these snapshot keys is pressed, the state of the patterns (which ones are armed versus unarmed) is instantly saved. While the snapshot key is held pressed, one can then change the state of

the patterns (using the keyboard, *not* the mouse) to change how the song plays. When the snapshot key is released, the original saved state of the patterns is restored.

Holding `Alt` will save the state of playing patterns and restore them when `Alt` is lifted.

The handling of `Alt` is often taken over by the window manager, so there could be a need to change these items to some other keys. For example, we have Fluxbox set up so that the `Alt` keys can be used for moving or resizing a window.

Holding `Right Ctrl` will queue a on/off toggle for a sequence when the loop ends. 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, as shown in the figure below.

Warning: 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, one can use the new **Menu** button in the main window to disable the menu, which makes those operations available. The `Super` key can also be used, if not already used by the desktop environment. Also available are some of the function keys, and, if available, the keypad keys. These can be configured in **File / Options / Keyboard** and **File / Options / Ext Keys**.

Note that queueing can also be used to turn a pattern *off* at the end of a pattern. Also note the "keep queue" functionality described below.

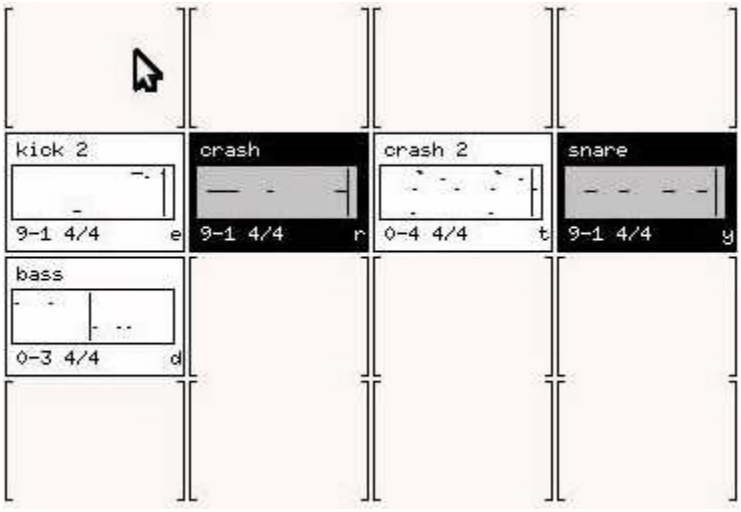


Figure 42: Pattern Coloration when Queued

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

Of course, the `Ctrl` key is used to manage the GUI (e.g. `Ctrl-Q` will unceremoniously quit the application), so one will usually want to change this key to something else in the **File / Options / Keyboard / Queue** field. The `Super` key (i.e. the `Mod4` or `Windows key`) is a good candidate to substitute for the `Ctrl` key, unless one has (like the author) configured the window manager to use the `Super` key to manipulate windows and applications *(laughter ensues)*. However, we have found that one can also use normal keys to enable queueing. For example, the minus key or the keypad's slash key can be used, with no noticeable flickering due to the keyboard repeat rate.

Pressing the "keep queue" hot-key (we like the backslash key) assigned in the "rc" file activates permanent queue mode until one click the regular (i.e. temporary) queue function hot-key, or changes the active (viewed) screen-set. After pressing the "keep queue" hot-key, individual pattern toggles occur at the end of the current pattern rather than immediately. While in queue mode, pressing a patterns hot-key, or clicking on the pattern, queues that pattern to change state at the beginning of the next loop. Keep queue mode is cancelled by pressing the normal queue hot-key.

This hot-key can be changed in the **File / Options / Keyboard / Keep queue** field. Also note the "queued-replace" functionality, described a bit later.

Note that there is also a "replace" hot-key, which is the left `Ctrl` key by default (which cannot work, because *Sequencer64* uses `Ctrl` as a way to access a lot of other functions, interfering with its use as a hot-key.

Replacement is a form of muting/unmuting. When the "replace" hot-key is pressed and held while clicking a pattern or clicking that pattern's hot-key, that sequence is unmuted, and all of the other sequences are muted. Again, this is a very inconvenient default, so an alternative key mapping should be set up in the "rc" file via the **File / Options / Keyboard** tab. Note that the "replace" functionality is a form of "solo". Also note that the "replace" functionality is also implemented via MIDI control, where the MIDI control can be activated, but then the user has to select the desired sequence.

Sequencer64 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 (click) the "keep queue" hot-key. This puts the application into "queue" mode. There is no visual indication (yet) of this mode.
3. Now press and hold the "replace" hot-key.
4. Click the desired pattern hot-key. Observe that it comes on, or stays on, and that the other playing patterns show the "queued" color (grey). At the end of the loop, they turn off, and the "replace" pattern is now soloed.
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 clear (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.

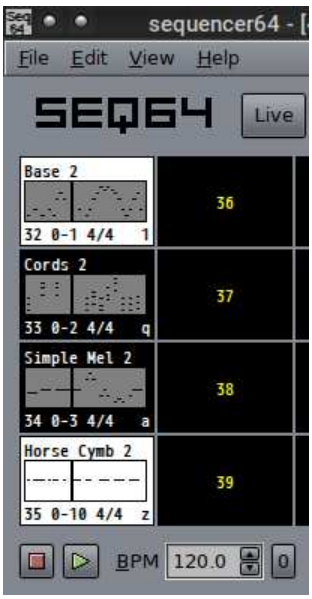


Figure 43: Queued-Replace (Queued-Solo) In Action

Before clicking 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.

Note that, 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 queue 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 clicking 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, queueing, and snapshots, see references [30] and [31].

3.2.3.2 Pattern Clicks

Left-clicking on a pattern-filled box will change its state from muted (white background) to playing (black background) when the sequencer is running.

By clicking and holding the left mouse button on a pattern, one can drag it to a new location on the grid. The box will disappear while dragged, and reappear in the new location when dropped. However, note that a pattern cannot be dragged if its Pattern Editor window is open.

Right-clicking a pattern will bring 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.

3.3 Patterns / Bottom Panel

The bottom panel of the Patterns window provides way to control the overall playback of the song.

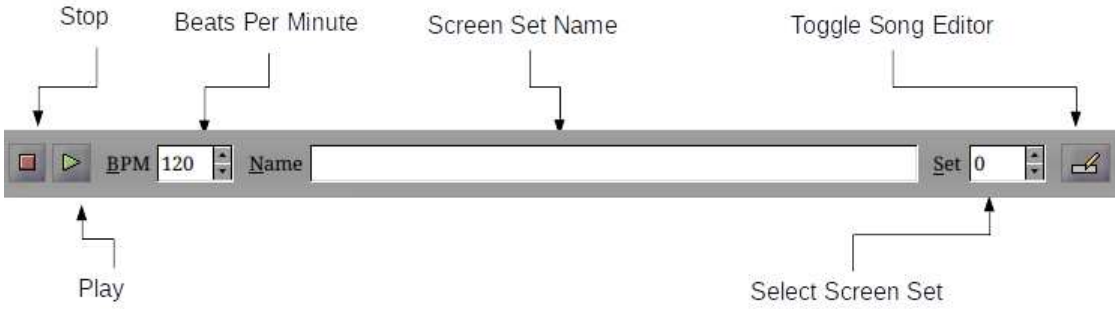


Figure 44: Patterns Panel, Bottom Panel Items

1. **Stop**
2. **Play**
3. **Pause**
4. **BPM**
5. **Tap Tempo**
6. **Name**
7. **Set**
8. **Toggle Song Editor**

1. Stop. The red square button stops the playback of the song and all its patterns. It is not clear if it also sends MIDI Off messages on all notes. The keystroke for stopping playback is the `Escape` character. It can be changed to `Space`, so that the space-bar then becomes a playback toggle key.

2. Play. The green triangular button starts the playback of the whole song. The keystroke for starting playback is the `Space` character.

3. Pause. A new feature is that the Play button can be used as a Pause button. When the Play button is clicked, the button icon changes to a Pause icon:

Figure 45: Patterns Panel, Pause Button

Note that a corresponding Pause key (by default, the period) has also been defined. The pause feature can be removed by rebuilding the application after configuring with the `--disable-pause` option.

4. BPM. The spin widget adjusts the "Beats Per Minute" or 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.

As a new feature, the precision of the BPM value can be set to 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. See section 9.4 "Sequencer64 "usr" File / User MIDI Settings" on page 122; it describes the precision and increment options more fully.

The following figure shows the appearance of the BPM field with different precision values:

Figure 46: Patterns Panel, BPM Precisions

5. Tap Tempo. This new control allows one to click it in time with a tune and 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. This function can also be performed using the keystroke defined in **File / Options / Ext Keys / Tap BPM**. It defaults to the `F9` key.

6. Name. Each of the 32 available screen sets can be given a name by entering it into this field.

7. Set. This spin widget selects the current screen set. The values in this field range from 0 to 31, and default to 0. Although this field looks editable, it is not.

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

4 Pattern Editor

The *Sequencer64 Pattern Editor* is used to edit and preview a pattern, as well as to configure its buss and channel settings.

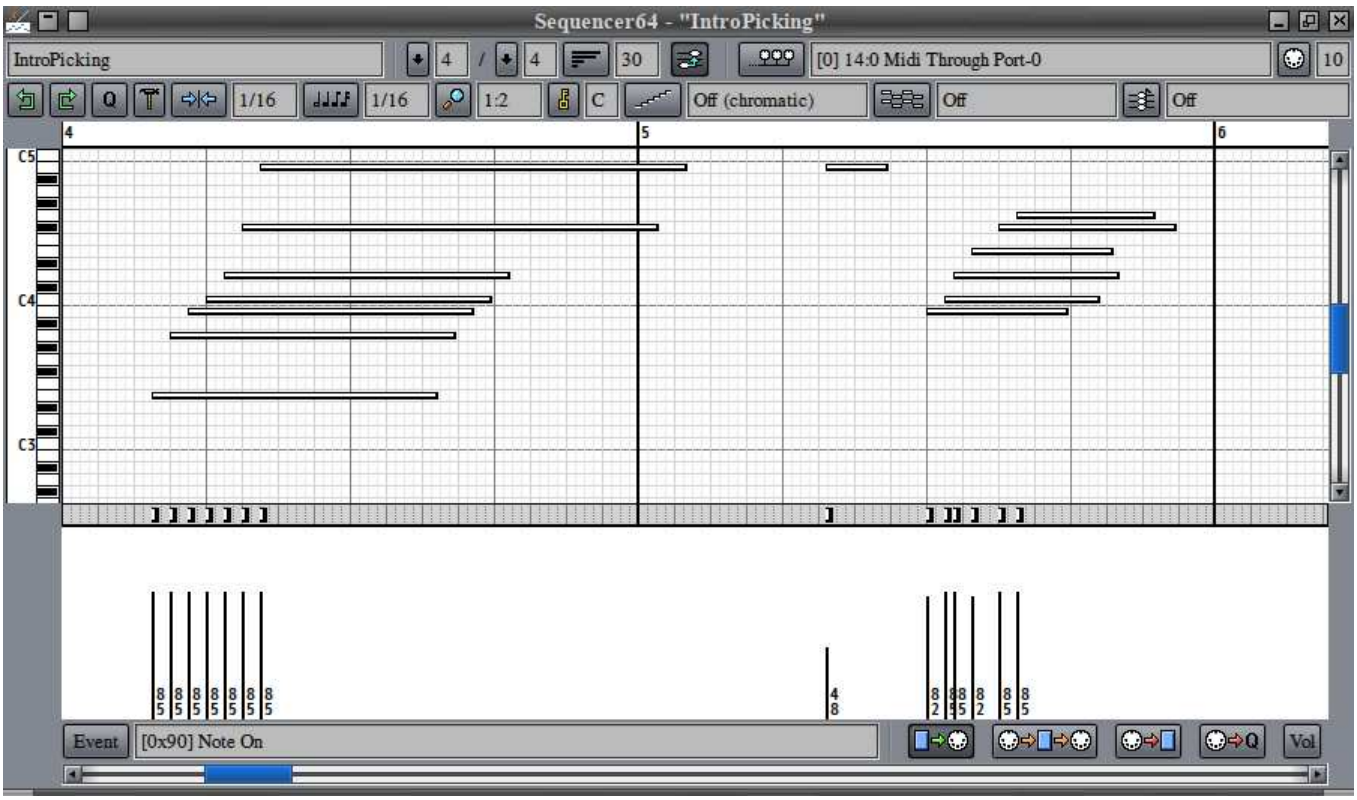


Figure 47: Pattern Edit Window

Not shown in this figure is the new **LFO** button in the bottom panel, but it is described later. This dialog is complex. For exposition, we break it into some common actions, a first panel, a second panel, a bottom panel, and a piano-roll/events section.

1. **First Panel**
2. **Second Panel**
3. **Piano-Roll/Events Panel**
4. **Bottom Panel**
5. **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. This is necessary to help the user key in a new pattern. Second, to add a note, one must press the right mouse button (the pointer changes to a pencil) and, while holding it, press the left mouse button. Or, to use a keystroke, click in the pattern editor, press `p` 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 that mode. Also remember that notes are drawn only with the length selected by the "notes" button near the top of the pattern window. There are some other tricks to modifying the new notes that are described later.

Also new with *Sequencer64*, and not in *Seq24*, is the automatic horizontal scrolling of 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.

Note that *Sequencer64* also provides a way to restart the progress bar within the pattern without resetting it to the beginning of the pattern. This new feature is called "pause". It can be accessed by the new

pause key (which defaults to the period character) or by the pause button, which appears when playback is underway. (Note that this feature can be disabled if the application is built via source code. Also note that it works only in ALSA mode at present.)

4.1 Pattern Editor / First Panel

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

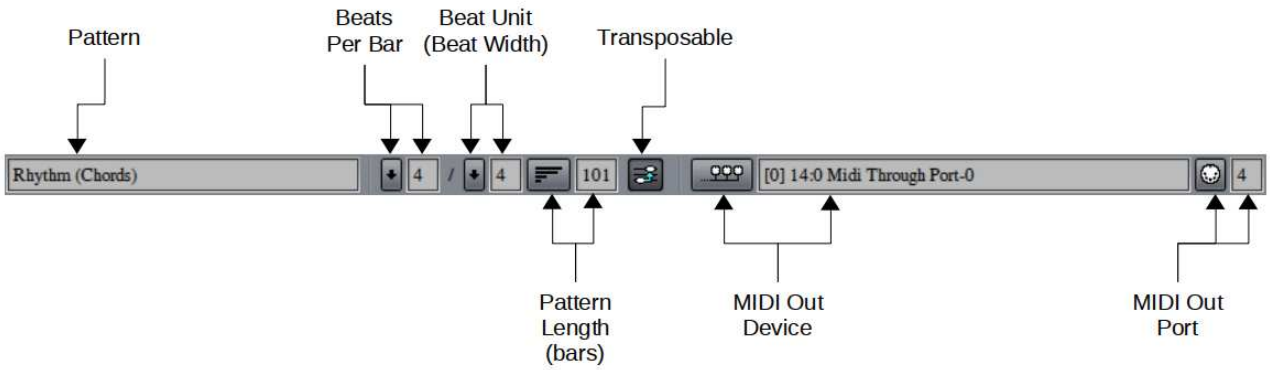


Figure 48: Pattern Editor, First Panel Items

In recent versions of *Sequencer64*, a "sequence-is-transposable" feature has been added, as shown above.

1. **Pattern Name**
2. **Beats Per Bar**
3. **Beat Unit (Beat Width)**
4. **Pattern Length**
5. **Transposable (toggle)**
6. **MIDI Out Device**
7. **MIDI Out Port**

1. Pattern Name. Provides the name of the pattern. This name should be short and memorable. It is displayed in the Patterns window, on the top line of its pattern box/slot. Unfortunately, there is no edit cursor in this field (not sure why yet), so one has to edit blind.

2. Beats Per Bar. Part of the time signature, and specifies the number of beat units per bar. The possible values range from 1 to 16.

3. Beat Unit (Beat Width). Part of the time signature, and specifies the size of the beat unit: 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 a pattern slot.

4. Pattern Length. Sets the length of the current pattern, in measures. The possible values range from 1 to 16, then 32, and 64. *However*, when opening or importing a non-*Sequencer64* MIDI tune, the length of each track will be used, and so other values are possible; they just cannot be set via the user-interface.

Note that bringing up a short sequence (one less than one measure or bar in length) in the pattern editor will adjust the sequence to pad it to the length of one measure. For example, a sequence containing just one program change will be padded to the size of a full measure. This adjustment makes it show progress

more smoothly in the main window when the sequence is playing. *Sequencer64* will, when it reads such a short sequence from a MIDI file (whether foreign or native to *Sequencer64*), pre-pad it to the length of a measure, so that it will always show smooth progress.

(It would sure be nice to have a value that represents "indefinite", so that the loop or pattern would be more like a track, and not be repeatable. Or perhaps add lengths of 128 and 200. The length of the longest track in our sample is 101 measures. Also nice would be a "one-shot" pattern, useful for live intro patterns, for example.)

5. Transpose Toggle. This item, if enabled in the build, allows the sequence to be transposed by the global transpose selection made in the song editor. 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.

6. MIDI Out Device (Buss). This setting specifies one of the 16 MIDI output busses provided by *Sequencer64*, or one of the MIDI devices set up in the computer. The settings look a lot like figure 40 "Existing Pattern, Right-Click Menu, MIDI Output Busses" on page 49.

7. MIDI Out Port (Channel). This settings select the MIDI output channel, or port. The possible values range from 1 to 16. If instruments are defined in the "user" configuration file to that device and channel, their names will be shown.

4.2 Pattern Editor / Second Panel

The second horizontal panel of the Pattern Editor provides a number of additional settings.

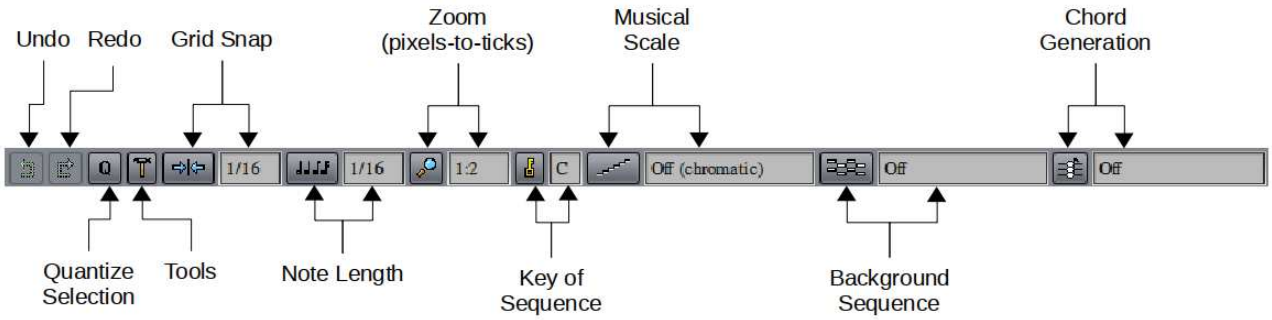


Figure 49: Pattern Editor, Second Panel Items

If *Sequencer64* is built with the new chord-generation option built in (the default), then the second panel is wider to make room for an addition user-interface item, shown at the right of the figure above.

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

11. **Chord Generation**

1. Undo. The Undo button will roll back any changes to the pattern from this session. It will roll back one change each time it is pressed. It is not certain what the undo limit (if any) is, however. 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. It is not certain what the redo limit is, however. There is no "Redo" key in the pattern editor.

3. Quantize Selection. Pressing this button will quantize 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.

Figure 50: Tools, Context Menu

1. **Select**
2. **Modify Time**
3. **Modify Pitch**

• **Select** provides two sets of selections for notes:

- **All Notes**, which selects all notes in the pattern; Note that `Ctrl-A` will also select all of the events in the pattern editor.
- **Inverse Notes**, which inverts the selection of notes.

Other event-selection actions are provided using the mouse in the piano roll:

- **Left Click**. Pressing the left button on a note or a event deselects all other notes or events, and selects the item clicked on.
- **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**. Pressing the left mouse button and dragging also lets one select ("lasso") multiple events and notes.
- **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 multiple notes in the selection.

There are many things that can be done with selected notes, as will be seen in the following paragraphs.

• **Modify Event Data** offers a way to change the event data (the lower pane of the pattern editor, the "data pane"). By left-dragging the mouse in the data pane across the value lines that are shown, the values are chopped or set to the height of the mouse pointer at each event. 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*.

• **Modify Time** offers two ways to tweak the timing of the selected note: **Quantize Selected Notes**, which quantizes the selected notes, the same way as the **Quantize** ("**Q**") button; **Tighten Selected Notes**, which is merely a less strict form of quantization.

• **Modify Pitch** has only one entry by default, **Transpose Selected** (not shown). Selecting the **Transpose Selected** entry brings up the following sub-menu:

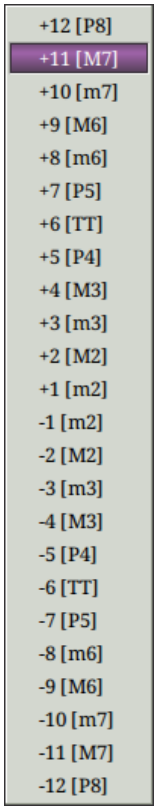


Figure 51: Tools, Transpose Selected Values

• If the user has selected a **Musical Scale** setting other than **Off**, then **Modify Pitch** has two entries: **Transpose Selected**, discussed above, plus another sub-menu, **Harmonic Transpose Selected**, which makes sure that all transpositions stay on the selected scale.

Note that one can also modify the pitch of selected notes by the left-click-drag action, or by moving the selection using the up-arrow or down-arrow keys.

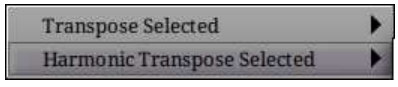


Figure 52: Tools, Two "Transpose" Menus

Remember that only the **Transpose Selected** entry is shown if the **Musical Scale** setting is **Off**. Selecting the **Harmonic Transpose Selected** entry brings up the following sub-menu:

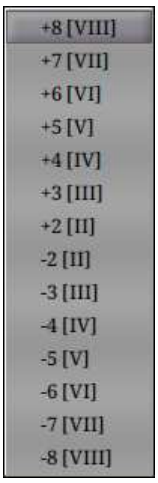


Figure 53: Tools, Harmonic Transpose Selected Values

Again, the harmonic-transpose option will not be available unless a scale has been selected.

5. Grid Snap. Grid snap selects where the notes will be drawn. The following values are supported: 1, 1/2, 1/4, 1/8, 1/16, 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.

6. Note Length. Note Length determines the duration of the inserted notes. Like the **Grid Snap** values, the following values are supported: 1, 1/2, 1/4, 1/8, 1/16, 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.

7. Zoom. Zoom is the relation between MIDI pixels and ticks, written as "pixels:ticks", where "ticks" is really the "pulses" in "PPQN". For example, 1:4 = 4 ticks per pixel. Supported values are 1:1, 1:2, 1:4, 1:8, 1:16, and 1:32, along with more new 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. This is the zoom of the Pattern Editor. As the right number goes higher, the effect is to zoom out, and show more of the pattern at once. In fact, it is probably better to list them as ticks (pulses) per pixel:

- 1 pulse per pixel
- 2 pulses per pixel (default value)
- 4 pulses per pixel
- 8 pulses per pixel
- 16 pulses per pixel
- 32 pulses per pixel (same as Song Editor)
- . . .
- 512 pulses per pixel

New: After one has left-clicked in the piano roll, the "z", "Z", and "0" can be used to zoom the piano-roll view. The "z" key zooms out, the "Z" key zooms in, and the "0" key resets the zoom to the default value. The zoom feature also modifies the time-line (measures indicator) and the data area. If, for some reason, the data area and piano-roll get out of sync, click on the horizontal scroll bar to force the views to redraw properly.

Note that the Song Editor, which now has zoom functionality (through the "z", "Z", and "0" keystrokes only), has a default resolution of 32 pulses per pixel, so, by default, it has 16 times the resolution of the Pattern Editor.

8. Key of Sequence. Selects the desired key for the pattern. The following scales are supported: C, C#, D, D#, E, F, F#, G, G#, A, A#, and B. Note that changing the **Key** will also shift the marked notes for the **Musical Scale** setting.

New: As of version 0.9.9.8, the key that a sequence is set to is now saved in the MIDI file along with the rest of the data for the sequence. **However**, it turns out that a change made to the key, scale, or background sequence in the sequence editor is saved in that editor, so that opening another sequence will apply the same settings to that sequence. This is an optional feature, now more rigorously supported, as noted below.

If the global-sequence-feature is enabled, and the user selects a different key, scale, or background sequence in the sequence editor, then *all* sequences will 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 sequences.

If the global-sequence-feature is *no*t enabled, and the user selects a different key, scale, or background sequence in the sequence editor, then only that sequence 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 sequence, as a SeqSpec meta event. The global-sequence-feature setting can be made in the "user" configuration file.

9. Musical Scale. Selects the desired scale for the pattern. 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, to make it easier to key all notes in-scale.
- When harmonic transposition is performed, the notes are shifted so that they remain in the selected scale.
- The exact notes that are considered "in-scale" depend also on the exact value of the **Key of Sequence** setting.

New: Originally, only the following scales were supported: Off, Major, and Minor. Now, the following scales are supported:

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

Please let us know of any mistakes found in the new scales. Also please note that the **Melodic Minor** scale is supposed to descend in the same was as the natural **Minor** scale, but there is no way to support that trick in *Sequencer64*.

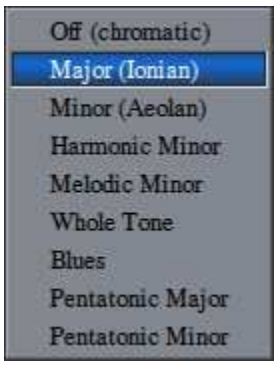


Figure 54: Scales Currently Supported in Sequencer64

One can select which **Musical Scale** and **Key** the piece is in, and *Sequencer64* will grey those keys on the piano-roll that are *not* in the selected scale for the selected key. This is purely a visual thing; a user can still add off-key notes. This effect is shown for the C Major scale in the following figure:

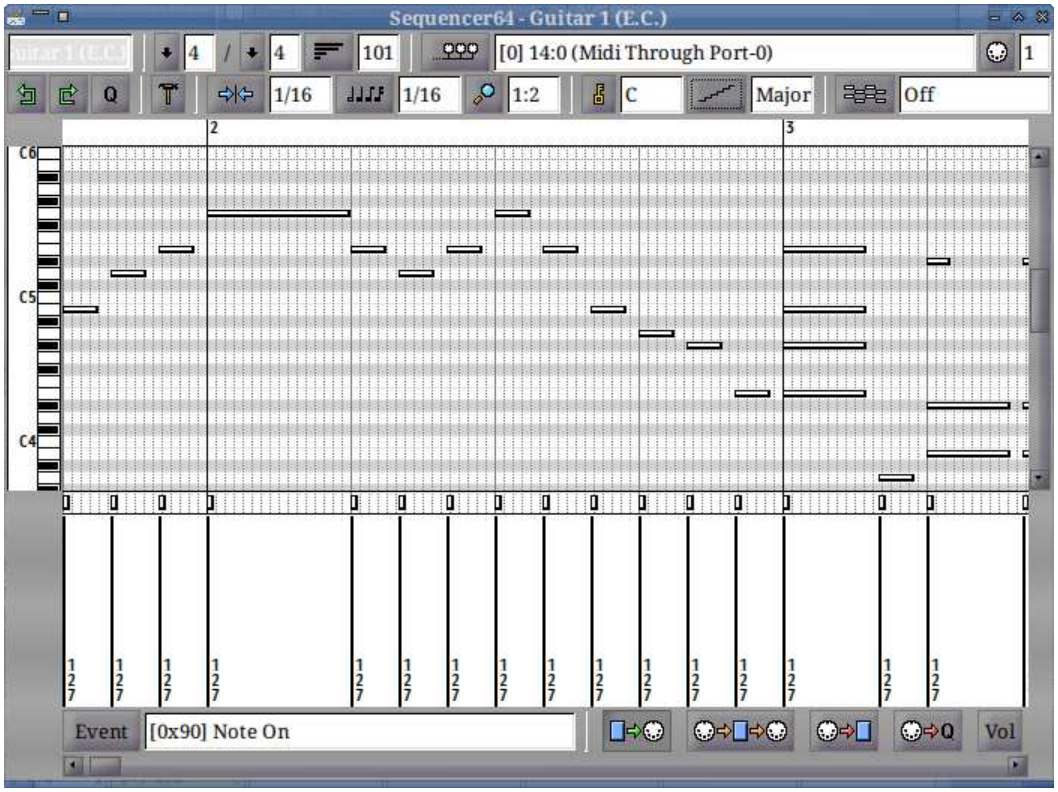


Figure 55: C Major Scale Masking

This feature makes it a bit easier to stay in key while playing and recording. Note that the scale will shift when a different **Key** is selected.

New: The scale that a sequence is set to is now saved in the MIDI file along with the rest of the data for the sequence. **However**, it turns out that a change made to the key, scale, or background sequence in the sequence editor is saved in the editor, so that opening another sequence will apply the same settings to that sequence. This is a feature. The feature had some quirks, which are fixed, and it is now an optional

feature. Also, the user has the option of applying the key/scale/background-sequence either globally (all sequences) or locally, per-sequence, with each sequence holding its key, scale, and background-sequence settings in SeqSpec meta events.

10. 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 **[0]**. 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 **0** entry, a menu like the following appears:

Figure 56: Sample Background Sequence Values

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. (Also note the orange selected notes and events in the following figure.)

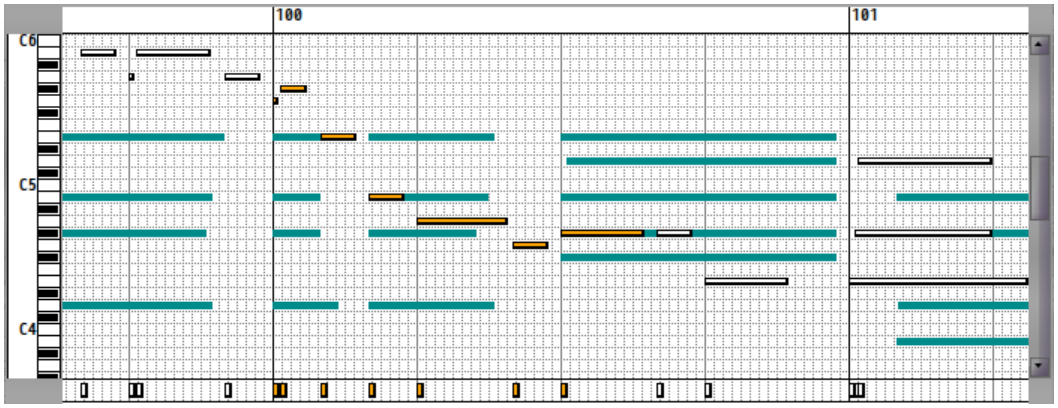


Figure 57: Background Sequence Notes

The dark cyan notes shown represent the rhythm pattern that was selected as the background pattern.

New: The background sequence that a sequence shows is saved in the MIDI file along with the rest of the data for the sequence. **However**, it turns out that a change made to the key, scale, or background sequence in the sequence editor is saved in the editor, so that opening another sequence will apply the same settings to that sequence. This is an optional feature, now more rigorously supported, as noted earlier.

11. Chord Generation. The ability to insert chords with one click has been added. This feature comes from user "stazed" and his *Seq32* project ([21]).

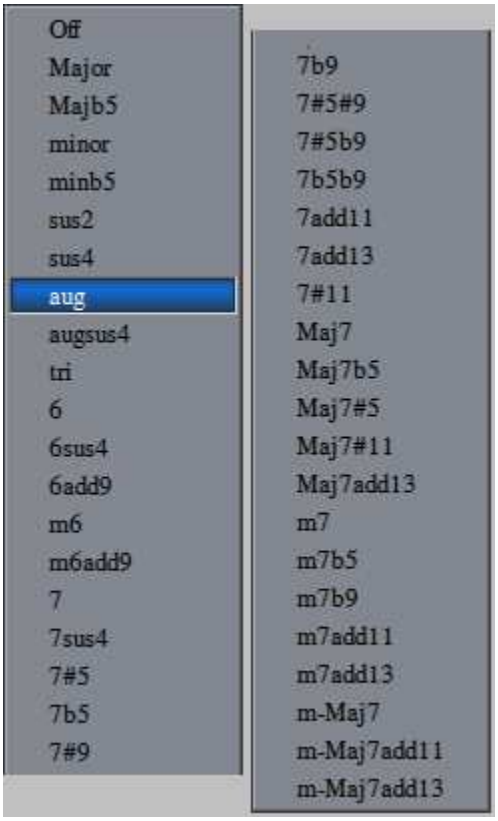


Figure 58: Chord Generation Menu

The figure shows the menu broken into two pieces. Once a value other than **Off** is selected, a left-click in 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 / Piano Roll

The piano roll is the center of the pattern (loop, track, sequence) editor. It is accompanied by a thin "event bar" (or "event area", or "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 3 or more buttons is practically a necessity for editing (though we have made *Sequencer64* more usable with some crummy trackpads now common on modern laptops, and also usable with keystrokes.) We tend to like the Logitech Marble Mouse, an ambidextrous USB trackball. It has four buttons, and we use the `contrib/scripts/marblemouse` script to set up the left small button as a middle button. The script merely makes the following call:

```
xmodmap -e "pointer = 1 8 3 4 9 6 7 2 5 10 11"
```

Editing is much easier after making that setting. Of course, keystrokes and additional mouse configuration have been added to make editing easier even without a good mouse. For example, one can page up and

down vertically in the piano roll using the Page Up and Page Down keys. One can go to the top using the Home key, and to the bottom using the End key. One can page left and right horizontally in the piano roll using the Shift Page Up and Shift Page Down keys. One can go to the leftmost position using the Home key, and to the rightmost position using the End key, And there are more keystroke actions to be described later.

Also, do not forget the note-step option. If one paints notes with the mouse, the note is previewed, and the note position advances with each click. If one paints notes via an external MIDI keyboard, the notes are painted and advanced, but they are not previewed. To preview them, click the **pass MIDI in to output** button to activate so that they will be passed to one's sound generator or software synthesizer.

4.3.1 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.

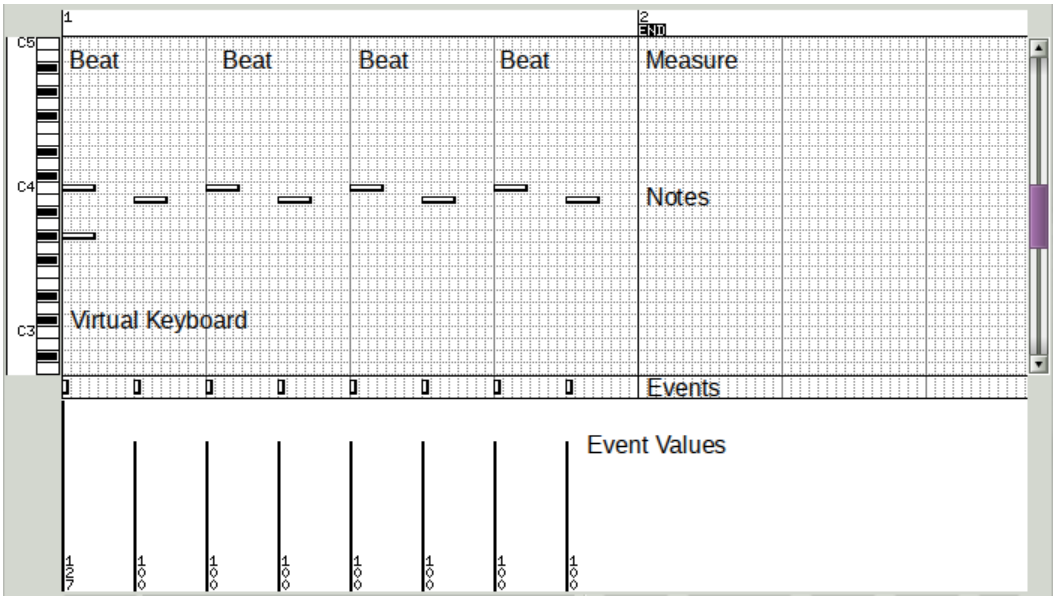


Figure 59: Pattern Editor, Piano Roll Items

1. **Beat**
2. **Measure**
3. **Virtual Keyboard**
4. **Notes**
5. **Events**
6. **Event Values**

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

2. Measure. The heavy vertical lines represent the measures 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 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. It can show marks to indicate off-scale notes, to make them easy to avoid. 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 anywhere in the virtual keyboard area toggles the display between the octave note letters and the MIDI note numbers (only every other one is displayed due to space, to avoid cramped numbering). The following figure shows both views, superimposed for comparison.

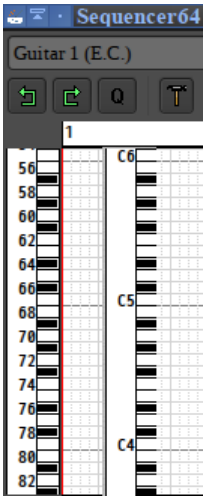


Figure 60: Pattern Editor, Virtual Keyboard Number and Note Views

4. Notes. Musical notes are indicated 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.

5. Events. Also known as the "events pane" or "events panel". The small (just a few pixels high) events strip shows discrete events, such as Note On and Note Off and their velocities, or various Controller items and their values. We recommend not editing or selecting events in that pane (feel free to disobey), but it is a good way to add events. Either left-click (to add one event), or left-click-drag horizontally (to add a series of events at the current note resolution.) 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.

6. Event Values. Also known as the "data pane" or "data panel".

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. 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.

4.3.2 Pattern Editor / Event Editing

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 actually very elegant in *Sequencer64* and in *Seq24*.

Note editing is a bit different with *Sequencer64*, since it requires two mouse buttons in many cases. There are some new laptop touchpads that really have only one mouse button, and use positioning to determine if the click is a left-click or a right-click. Two-fingered actions are devoted to scrolling, so that there is

no way to generate a Linux middle-click. One solution is to use a four-button USB trackball configued with an easy middle-button setup. It's easier than a touchpad, anyway. So we've coded up a couple of solutions to this middle-click problem.

Note that, if only a middle-button is needed, `Ctrl-left` will simulate that button.

New: There is a feature to allow the Mod4 (the Super or Windows) key to keep the right-click in force even after it is released. See 2.2.5.5. Basically, pressing Mod4 before releasing the right-click that allows note-adding, keeps note-adding in force after the right-click is released. Now notes can be added at will with the left mouse button. Right-click again to leave the note-adding mode.

New: Another way to turn on the paint mode has been added. To turn on the paint mode, first make sure that the piano roll has the keyboard focus by left-clicking in it, then press the `p` key while in the sequence editor. This is just like pressing the right mouse button, but the draw/paint mode sticks (as if the Mod4 mode were in force). To get out of the paint mode, press the `x` key while in the sequence editor, to "x-scape" (get it? get it?) from the paint mode. These keys also work while the sequence is playing.

The `p` and `x` keys also works in the small event strip just above the white data area. The Mod4-right-click feature does not yet work in that aread of the user interface, but the `p` key does.

4.3.2.1 Editing Note Events

The Piano Roll pane provides for a quite sophisticated set of note-editing actions. Not only is there a native mouse-interaction mode, but there is a "fruity" mouse-interaction mode that works more like the applications "Fruity Loops", it's follow-on "FL Studio', and its Linux look-alike, "LMMS".

1. Fruity Mode. At some point, we will add a section detailing the usage of the "fruity" mode of mouse-interaction.

Please study the following paragraphs carefully, ideally while trying them out in *Sequencer64*.

2. Enter Draw Mode. In the note (grid/roll) panel, **holding** down the **right** mouse button will change the cursor to a pencil and put the editor into "draw" mode, also known as "note-adding" or "paint" mode. To exit the draw mode, release the right mouse button, and the cursor will turn back into an arrow.

Another way to enter paint mode is to make sure the piano roll has focus (left click in it), and then press the `p` key. To exit the draw mode, press the `Shift-p` key.

3. Add Notes. Then, while still **holding** the **right** mouse button, click the **left** mouse button to **insert** new notes. Many people find this combination strange at first, but once one gets used to it, it becomes a very fast method of note manipulation. An new option is to hold the Mod4 key while releasing the right button, which keeps the mouse in draw mode. Another new option is to press the `p` to enter draw mode, and stay in it until `Shift-p` is pressed. Note that this click will add a single note, and the length of the note will be that specified in the note-length setting (e.g. "1/16").

To increase the number of notes, keep dragging the mouse (with both buttons held). It can be dragged rightward, leftward, upward, and downward. Dragging left or right adds new notes, while dragging upward or downward moves the current note to a different pitch. We call this the "auto-note" feature. Please note that the auto-note feature does *not* work with the chord-generation feature.

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.

Thus, one can, with some care, draw a nice chorded sequence. Adjustments to it can be made afterward.

4. Select Notes. 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.

To select a single note, simply **left click** on it. The selected note will turn orange.

To select multiple notes, perform a **left click drag** to form a selection box that intersects (even partly) the desired notes. Once the mouse is released, all of the desired notes should be orange.

To add more notes to a selection of notes, move to an unselected note and perform a **ctrl left click drag** to form a selection box that intersects (even partly) the desired notes. Once the mouse is released, all of the desired notes should be orange. 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.

Pressing the `Ctrl-A` key will select all of the events in the pattern editor.

The **Tools** button described in section 4.2 "Pattern Editor / Second Panel" on page 58can also be used to modify selections.

Once one or more notes are selected, they can be modified in time, pitch, or length.

5. 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.

6. Move Notes in Pitch. To move notes in pitch, once selected, grab one of the notes in the selection and drag it upward or downward. **New:** Also, since a selection is in force, the Up and Down arrow keys can also be used to change the pitch of every note in the selection. The smallest unit of pitch change is one MIDI note value.

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!

7. Move Notes in Time. To move notes in time, once selected, grab one of the notes in the selection and drag it leftward or rightward. **New:** 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.

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.

(There is another feature of the arrow keys when no selection is made, that supposedly moves an "origin" for playback, but we haven't been able to figure out exactly if it does anything.)

8. Change Note Length. Pressing the **middle** mouse button ***or*** pressing the **ctrl left** mouse button in tandem, while the pointer is hovered over a selected note, will let one change the length of a selected note. If more than one note is selected, then the length of all selected notes is changed.

Once a selection of notes is made, one can use the shift-middle-click-drag (true?) or ctrl-left-click-drag sequence over the selected notes to draw a box beyond the extent of the notes. When the mouse is released, each of the events is moved and lengthed to be proportionally longer to fit exactly within the box one drew. This feature is called *event stretch*.

If the box that was drawn was shorter than the original extent of the notes, then the notes move and shrink proportionally to occupy the smaller box. This feature is called *event compression*.

Warning: If one reduces or increases the length of a note selection by too much, the note or notes will "wrap-around" to the end of the sequence boundary and grow more from the beginning of the sequence. It is not clear if this new note has an ending time that is less than its beginning time. If it happens, one probably ought to undo it.

Note that notes can be shortened below the default note length by event compression. Note that there is currently no way to change the length of the note using a keystroke.

9. 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`), Paste (`Ctrl-V`), and "drop" (`Enter`) 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. However, once `Ctrl-V` is struck, then one must *move* the mouse pointer to see where to paste the events, or move it with the arrow keys. An orange box representing the selected-and-copied notes appears, and the user should move the box (note) to the desired location and then left-click.

(Warning: We've had temporary issues where the selection box flickers, and this seems to be due to updates in the graphics library used by Sequencer64. This issue might depend on the Linux distro one uses.)

Additionally, one can move the orange box using the arrow keys, to the desired location, and then hit the `Enter` key to drop the notes at that location.

Finally, note that selected notes that are cut or copied can then be pasted into *other* pattern editor dialogs; that is, they can be pasted into other sequences.

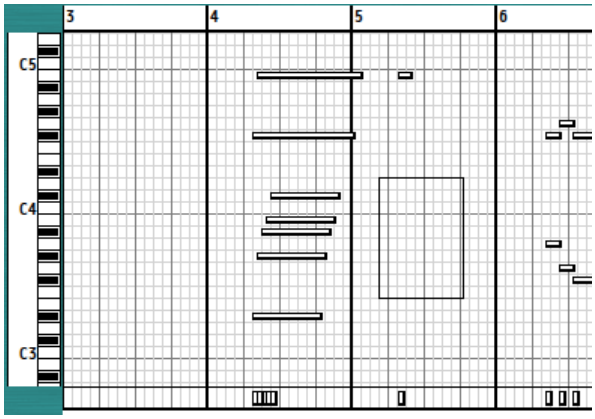


Figure 61: Piano Roll, Paste-Box for Cut Notes

Note that the selection box is now orange, not black. Move this box to where pasting is desired, and

left-click. The moved notes appear, still selected, and they can then be moved further, if desired, by using the arrow keys, or cut and move them again.

For the appearance of selected events (orange), see figure 62 "Piano Roll, Selected Notes and Events" on page 71.

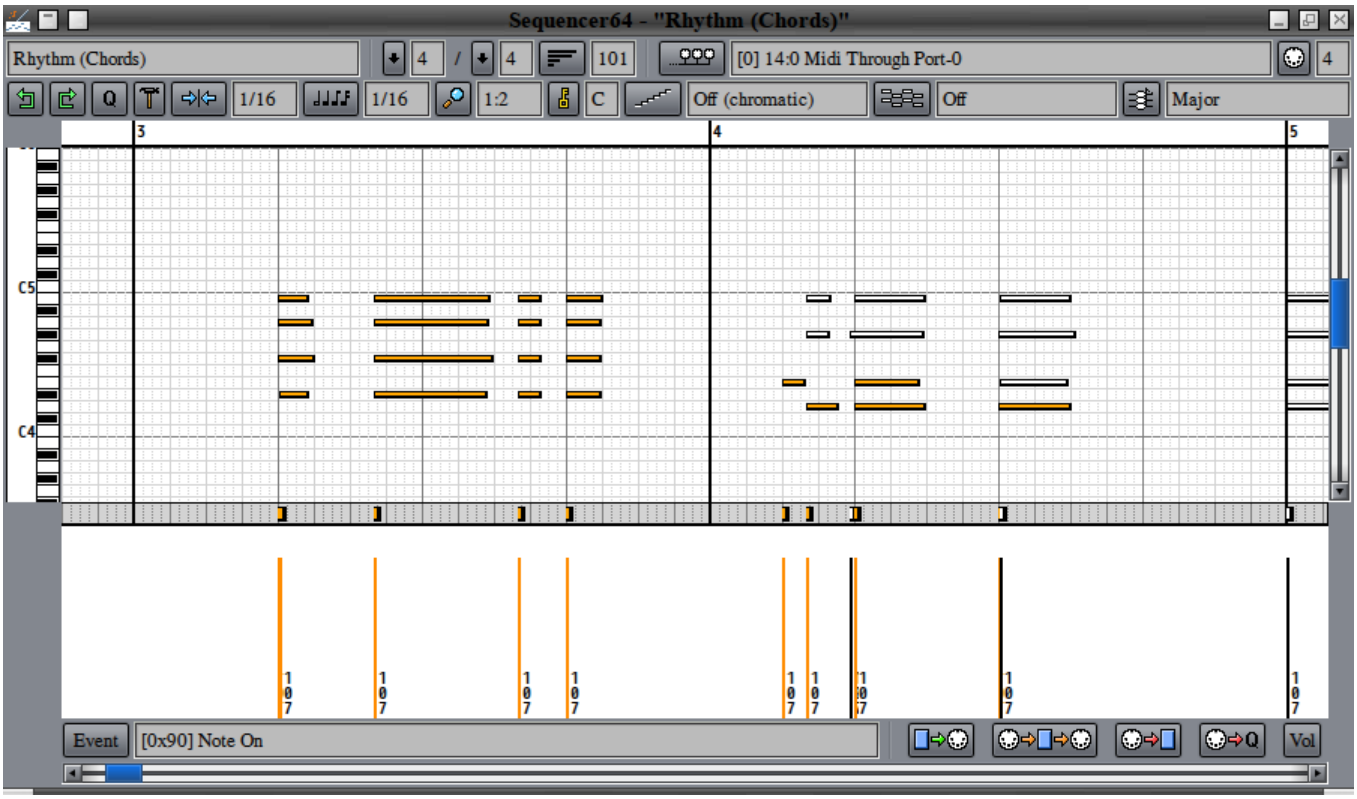


Figure 62: Piano Roll, Selected Notes and Events

New: The selection, shown in figure 62 "Piano Roll, Selected Notes and Events" on page 71, illustrates the new style of event selection, which colors the data bars as well as the event and note bars. This selection was made by first selecting one set of events by the left-click-drag action, then selecting more events by holding the Ctrl key for the next left-click drag action. The second selection left out some events, which are thus still shown as black bars in the data area.

As an aside, note that the event strip is gray. *Sequencer64* now shows this strip as gray when the selected event (the **Event** button) is Note On, Note Off, or Aftertouch. This color is purely a reminder that moving these events can really screw up the notes, for example by moving Note Off to before the Note On event.

4.3.2.2 Editing Other Events

Note On and *Note Off* events (and other events) can appear as small squares in the event strip, along with a black vertical bar with a height proportional to the velocity of the note event, plus a numeric representation of that value. Note events do not need to be inserted in the event strip. (*Note events can be inserted there, but they end up as short events of the lowest possible note, 0 or C1, and they don't have a Note Off event. So don't do that!*)

Other event types 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. The place the mouse cursor in the event strip. Right-click to make the drawing pencil 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 should appear.

Should one want more of the same event, continue to hold both buttons and drag the mouse. One event should appear at each beat position (e.g. at each 16th note position) that is crossed.

To move the event(s) to a different spot, select it or them via the left button. Then drag it or them to where one wants them. it is currently not possible to move them to positions smaller than the beat size. The work-around is to temporarily reduce the beat size, but this requires caution.

Once the event positions are set, the next step is to modify the data values of the events.

The event value (data) editor (directly under the event strip) is used to change note velocities, channel pressure, control codes, patch select, etc. Just left-click+drag the mouse across the window to draw a line. The values will match that line. middle-click+drag and right-click+drag also draw the value line.

Bug: Sometimes the editing of event values in the event data section will not work. The workaround is to do a `Ctrl-A`, and the click in the roll to deselect the selection; that makes the event value editing work again.

Any events that are selected in the piano roll or event strip can have their values modified with the mouse wheel.

4.3.2.3 Editing Note Events the "Fruity Way"

This mode is a lot different, and we have yet to do the exhaustive testing needed to understand how this mode works. Input from actual users of this mode would be welcome.

4.4 Pattern Editor / Bottom Panel

The bottom horizontal panel of the Pattern Editor provides for selecting events for viewing and edition, and the MIDI playback, pass-through, and recording options of *Sequencer64*.

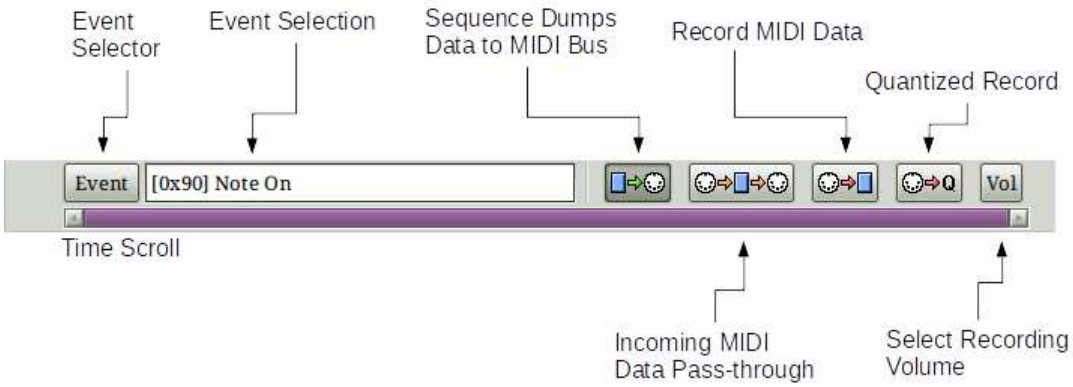


Figure 63: Pattern Editor, Bottom Panel Items

1. **Event Selector**
2. **Event Selection**
3. **LFO** (if built with LFO support; not shown)

4. **Time Scroll**
5. **Data To MIDI Buss**
6. **MIDI Data Pass-Through**
7. **Record MIDI Data**
8. **Quantized Record**
9. **Select Recording Volume**

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

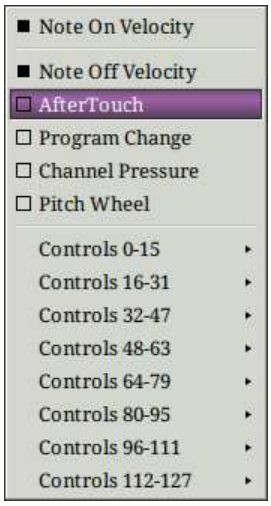


Figure 64: Pattern Editor, Event Button Context Menu

The sub-menus of this context menu show 128 controller messages, so we won't try to show all of them here.

These sub-menus can be modified, as far as we know, by editing the file

```
$HOME/.config/sequencer64/sequencer64.rc
```

(legacy mode: `$HOME/.seq24usr`), to make it match one's instrument. See section 9 "Sequencer64 "usr" Configuration File" on page 113.

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

New:

3. LFO. If *Sequencer64* is built with the `--enable-lfo` option, then a low-frequency oscillator functionality is built so that data events can be modulated by some rudimentary wave functions. This feature is now enabled by default. By clicking on the **LFO** button or using the `Ctrl-L` key, the following window appears:

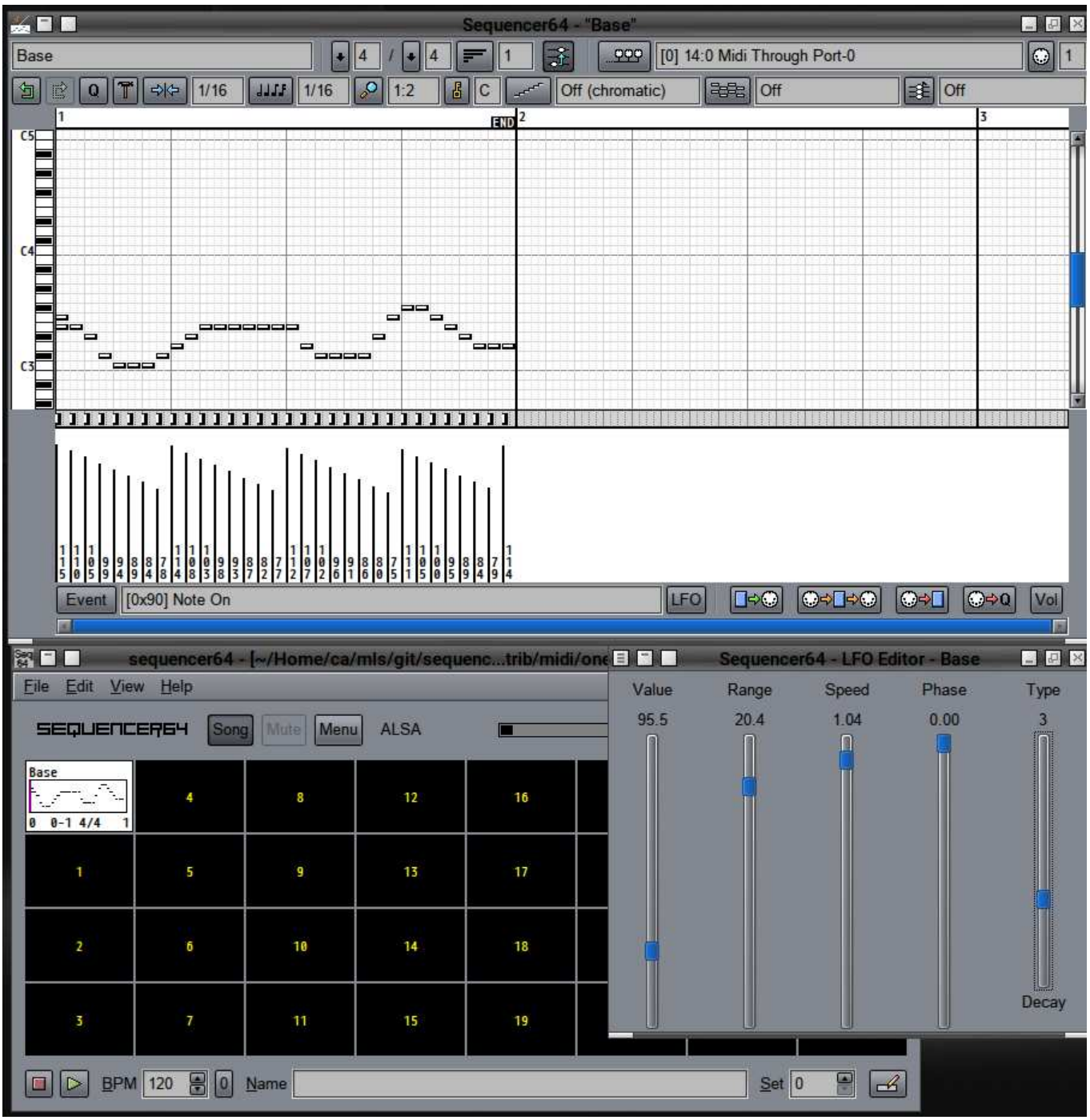


Figure 65: Pattern Editor, LFO Support

Note the controls in this window:

1. **Value**: Provides a kind of DC offset for the data value. Starts at 64, and ranges from 1 to 127.
2. **Range**: Controls the depth of modulation. Starts at 64, and ranges from 1 to 127.
3. **Speed**: Indicates the number of periods per pattern (divided by beat width, normally 4). For long patterns, this parameter needs to be set high, to even show an effect. It is also subject to an 'anti-aliasing' effect in some parts of the range, 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 four periods of the wave.

4. **Phase**: 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. **Wave Type**: Selects the kind of wave to use for the LFO:
    1. **Sine wave**.
    2. **Ramp (up) sawtooth**.
    3. **Decay (down) sawtooth**.
    4. **Triangle wave**.

We may have more to explain about this dialog at some point. For now, try it out on the file `one-measure.midi`, and be sure to hover over each control to see the tooltips. Note that it works best with short patterns.

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

5. Data To MIDI Buss. Activating this button will cause the pattern to be output to the MIDI output buss, which will normally be connected to a software or hardware synthesizer, to be heard. Generally, this control should always be activated.

6. MIDI Data Pass-Through. Activating this button will route incoming MIDI data through *Sequencer64*, which will then write it to the MIDI output buss.

7. Record MIDI Data. Activating this button will route incoming MIDI data into *Sequencer64*, which will then save the data to its buffer, and also display the new information (notes) in the piano roll view.

8. Quantized Record. Activating this button will also cause MIDI data to be recorded, but it will be quantized on the fly before recording it.

9. Vol. This button allows controlling the volume of the recording. The velocity of the notes will be set to the selected value upon recording.

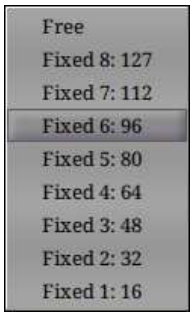


Figure 66: Pattern Recording Volume Menu

The values provided are:

- Free (record incoming volumes),
- Fixed 8, Fixed 7, Fixed 6, Fixed 5, Fixed 4, Fixed 3,
- Fixed 2, and Fixed 1.

These values correspond to MIDI volume levels from 127 down to 16, as shown in the figure.

One thing fixed in the 0.90.x version is the ability to store MIDI note-on events with the actual velocity provided by the MIDI keyboard used to generate the notes. Previously, even in *seq24*, the "Free" option in the "Vol" menu option did not work. This is fixed.

4.5 Pattern Editor / Common Actions

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

4.5.1 Pattern Editor / Common Actions / Scrolling

Let us describe the actions that can be performed with a scroll wheel, or with the scrolling features of multi-touch touchpads. There are three major scrolling actions available when using mouse scrolling, with the mouse hovering in the piano-roll area:

- **Vertical Panning (Notes Panning)** Using the vertical scroll action of a mouse or touchpad moves the view of the sequence/pattern notes up and down. One can also click in the piano roll, and then use the `Page-Up` and `Page-Down` keys to move the view up and down in pitch.
- **Horizontal Panning (Timeline Panning)** Holding the Shift key, and then using the vertical scroll action of a mouse or touchpad moves the view of the sequence/pattern time forward and backward. One can also click in the piano roll, and then use the `Shift Page-Up` and `Shift Page-Down` keys to move the view left and right in time.
- **Horizontal Zoom (Timeline Zoom)** Holding the Ctrl key, and then using the vertical scroll action of a mouse or touchpad zooms the view of the sequence/pattern time to compress it or expand it. One can also click in the piano roll, and then use the `z` , `Z` , and `0` to change the timeline zoom.

The actions of this scrolling are surprisingly smooth and fast. If an event is selected in the piano-roll area or the (thin) event area, then the scrolling action 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.

4.5.2 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.

Sequencer64 also provides the Ctrl-W key to close the pattern editor window.

However, be aware that this convention does not apply to the other application windows of *Sequencer64*.

5 Song Editor

The *Sequencer64 Song Editor* is used to combine all of the patterns into a complete tune. It works by showing one row per pattern/loop/sequence in numbered columns, and the placement of each pattern at various musical bars in the song. In *Sequencer64* parlance, the Song Editor creates a *performance*.

New: As an option in the `[user-interface]` section of the "user" configuration file, two song editor windows can be brought onscreen, as a convenience for arranging projects with a large number of

sequences/patterns. It also provides the "song mode" of *Sequencert64*, as opposed to the "live mode" provided by the Patterns Panel, when *Sequencer64* is running in ALSA mode. (In JACK mode, the live versus song mode is controlled by the JACK start-mode flag.)

When the Song Editor has the focus of the application, in ALSA mode, it takes over control from the Patterns Panel. The Song Editor then controls playback. Once playback is started in the Song Editor, some actions in the Patterns Panel no longer have effect, effectively disabling live mode. 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 playing in the Song Editor, or not. If one tries to change the muting using a hot-key (or even a click) 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*.

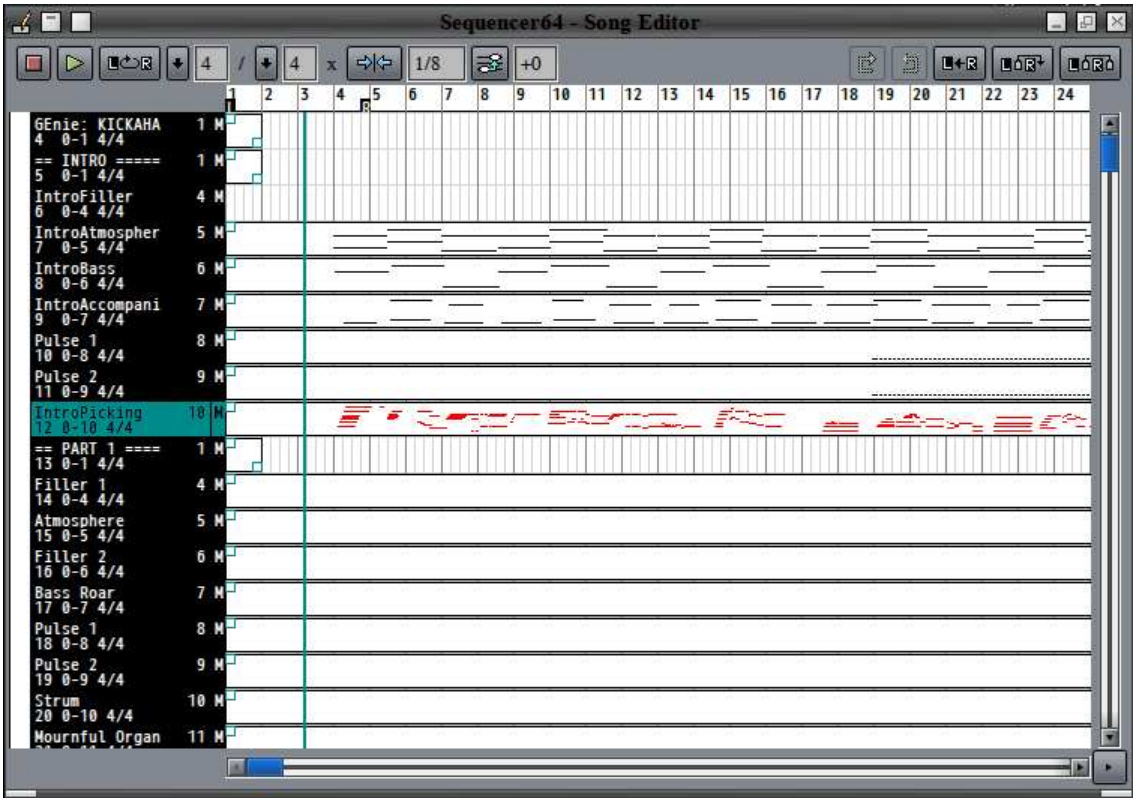


Figure 67: Song Editor Window

New: There are some new features for the song editor, as seen above and in the following figure:

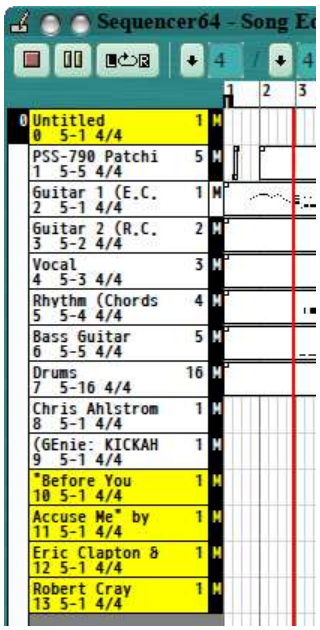


Figure 68: Song Editor Window, New Features

- Toggling of the mute state of multiple patterns by holding the Shift key while left-clicking on the M or a pattern name;
- The new Pause button functionality;
- The optional coloring (about 7 color values are selectable) and thickening of the progress bar.
- A new Redo button (not shown).
- A new Transpose button (not shown).
- Red coloring of events for patterns that are not transposable, such as drum tracks.

New: This dialog (in *Sequencer64*) shows any empty patterns highlighted in yellow. An empty pattern is one that exists, but contains only meta information, and contains no MIDI events that can be played. For example, some tracks just serve as name tracks or information tracks.

The Song Editor is not too complex, but for exposition, we break it into the top panel, the bottom panel, and the rest of the window.

New: There are still more features of the song editor in pending version 0.9.15, as seen in the following figure:

Figure 69: Song Editor Window, Transpose Song

Notice the additional button that allows the whole song (except for exempt, non-transposable sequences) to be transposed up or down by up to an octave in either direction.

5.1 Song Editor / Top Panel

The top panel provides quick access to song-playback actions and configuration.

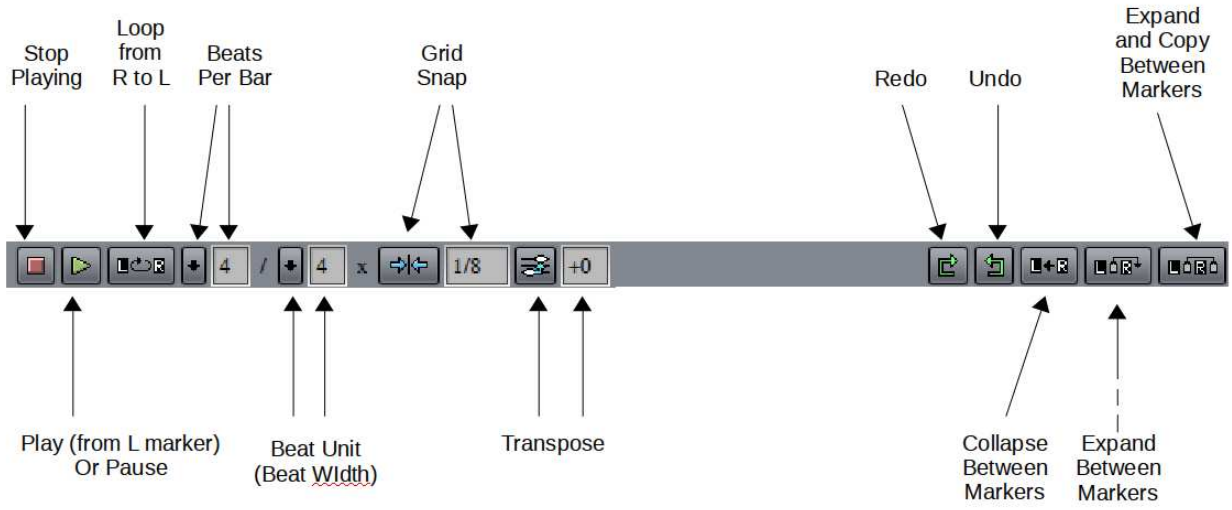


Figure 70: Song Editor / Top Panel Items

1. **Stop**
2. **Play**
3. **Play Loop**
4. **Beats Per Bar**
5. **Beat Unit**
6. **Grid Snap**
7. **Redo**
8. **Undo**
9. **Collapse**
10. **Expand**
11. **Expand and copy**

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 keystroke for starting playback is the `Space` character. The keystroke for stopping playback is the `Escape` character. The keystroke for pausing playback is the `Period` character, which currently works only in ALSA mode.

3. Play Loop. Activate loop mode. When Play is activated, play the song and loop between the **L marker** and the **R marker**. This button is a state button, and its appearance indicates when it is depressed, and thus active. If this button is deactivated during playback, then playback will continue past the **R marker**.

4. Beats Per Bar. Part of the time signature, and specifies the number of beat units per bar. The possible values range from 1 to 16.

5. Beat Unit. Also called "beat width". Part of the time signature, and specifies the size of the beat unit: 1 for whole notes; 2 for half notes; 4 for quarter notes; 8 for eight notes; and 16 for sixteenth notes.

6. Grid Snap. Grid snap selects where the patterns will be 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: 1, 1/2, 1/4, 1/8, 1/16, 1/32, and 1/3, 1/6, 1/12, and 1/24.

7. Redo. The Redo button will reapply the last change that was "undone" by the Undo button. It will be inactive if there is nothing to redo.

8. Undo. The Undo button will roll back the last change in the layout of a pattern. Each time it is clicked, the most recent change will be undone. It will roll back one change each time it is pressed. It is not certain what the undo limit is, however. It will be inactive if there is nothing to undo.

9. 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 wiped out, and the song material after the **R marker** is moved leftward to the **L marker**. Collapsing occurs in all tracks present in the Song Editor.

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

11. 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.

5.2 Song Editor / Arrangement Panel

The arrangement panel is the middle section shown in figure 67 "Song Editor Window" on page 77. It is also known as the "piano roll" of the song editor. Here, we zero in on its many features.

Keystrokes and additional mouse configuration have been added to make editing easier even without a good mouse. For example, one can page up and down vertically in the arrangement panel using the Page Up and Page Down keys. One can go to the top using the Home key, to the bottom using the End key. One can page left and right horizontally in the arrangement panel using the Shift Page Up and Shift Page Down keys. One can go to the leftmost position using the Home key, and to the rightmost position using the End key,

The following figure is taken from a conventional MIDI file, imported, with a few long tracks, rather than a large number of smaller patterns. In other words, the patterns used here are very long, and used only once in the song. (We will provide an example that shows off *Sequencer64*'s pattern features better, at some point.)

Please note that, if playback is started with the Song Editor as the active window, then the pattern boxes in the patterns panel will show as armed/unarmed (unmuted/muted) depending upon whether or not the pattern is shown as playing (or not) at the current playback position in the Song Editor piano roll.

The following figure highlights the main features of the center panel of the song editor.

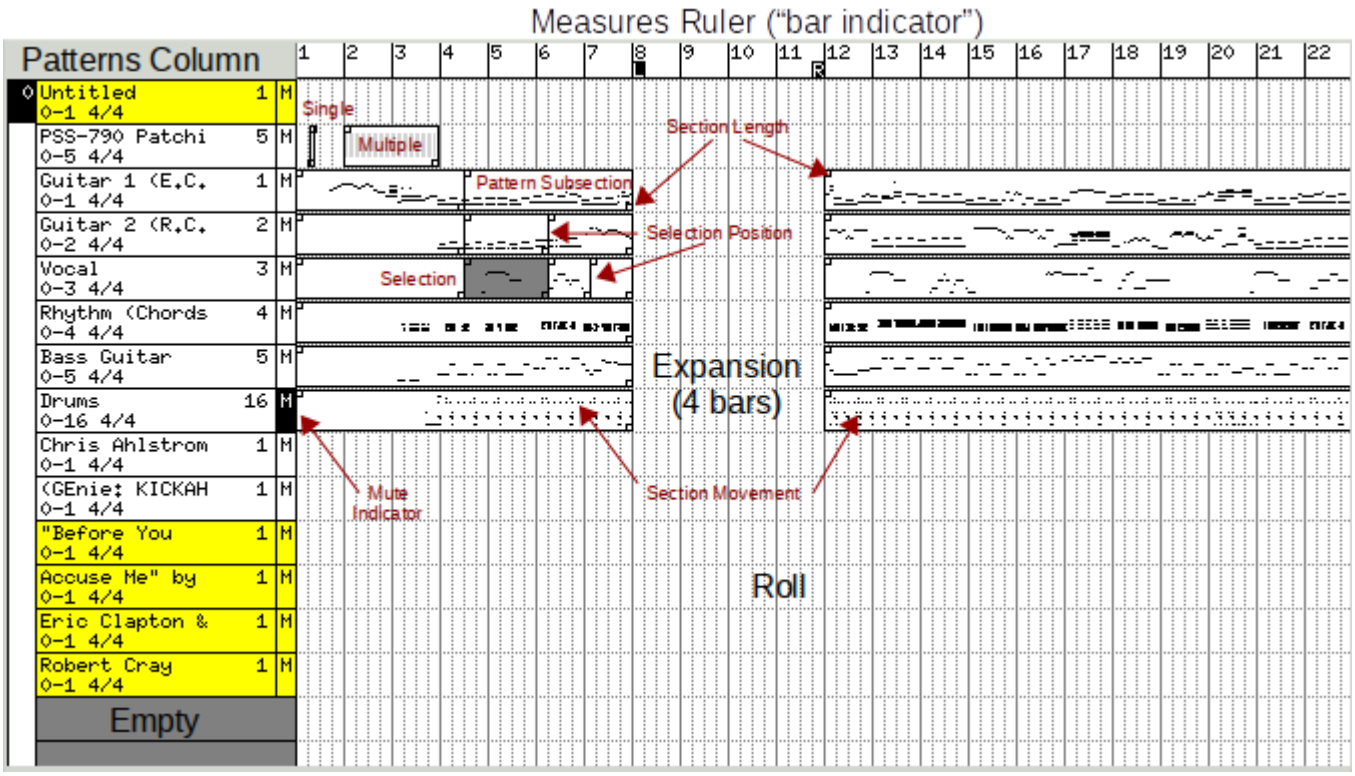


Figure 71: Song Editor Arrangement Panel, Annotated

One new feature of the song editor is that, if the new Transpose feature is built into *Sequencer64*, any patterns that are marked as exempt from transposition (common with drum tracks) have their events shown in red instead of black.

Figure 72: Song Editor for Non-Transposable Patterns

The measures ruler (measures strip) consists of a *measures ruler* (bar indicator) at the top, a numbered patterns column at the left with a muting indicator, and the grid or roll section. There are a lot of hidden details in the arrangement panel, as the figure shows. Here are the main sections we will deal with:

1. **Patterns Column**
2. **Piano Roll**
3. **Measures Ruler**

These items are discussed in the following sections.

5.2.1 Song Editor / Arrangement Panel / Patterns Column

Here are the items to note in the patterns column:

1. **Number**. Not yet sure what the number on the left means. The number of the screen set?

2. **Title**. The title is the name of the pattern, for easy reference.
3. **Channel**. The channel number appears (redundantly) at the right of the title.
4. **Buss-Channel**. This pair of numbers shows the MIDI buss number used in the pattern and the channel used for the pattern.
5. **Beat/Measure**. This pair of numbers is the standard time-signature of the pattern.
6. **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) will mute/unmute 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.
7. **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" button toggles the muting (arming) status of the track.

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 3.2.2 "Pattern" on page 45. Recall that this context menu has the following entries: **Edit...**, **Event Edit...**, **Cut**, **Copy**, **Song**, **Disable Transpose**, and **MIDI Bus**.

5.2.2 Song Editor / Arrangement Panel / Piano Roll

The "Piano Roll" section of the arrangement panel is where patterns or subsections are inserted, deleted, shrunk, lengthened, or moved. Here are the items to note in the annotated Piano Roll area shown in figure 71 "Song Editor Arrangement Panel, Annotated" on page 81:

1. **Single**. In the diagram, under the word "Single", is a very small pattern. It is small because it consists only of some MIDI Program Change messages meant to set the programs on a Yamaha PSS-790 keyboard.
2. **Multiple**. This item is the same pattern as in "Single", but dragged out for multiple repetitions, simply to show how even the shortest patterns can be replicated easily.
3. **Pattern Subsection**. Middle-clicking 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. (Note that, in the Song Editor, a middle-click *cannot* be simulated by ctrl-left-click.)
4. **Selection Position**. A selection position is a marker that divides a pattern into two pieces, called *pattern subsections*. This makes it easy to select smaller portions of a pattern for editing or deleting. It is especially useful for making holes in a pattern. There may be other uses of a selection position that we have not yet discovered.
5. **Selection**. By clicking inside a pattern or a pattern subsection, it darkens (gray) to denote that it is selected. A pattern subsection can be deleted by the Delete key, copied by the `Ctrl-C` key, and then inserted (one or more times) by the `Ctrl-V` key. When inserted, each insert goes immediately after the current item or the previous insertion. The same can be done for whole patterns.

6. **Section Length**. Looking closely at the diagram where the arrows point, small squares in two corners of the patterns can be seen. By grabbing that square with a left-click, the square can be moved horizontally to either lengthen or shorted the pattern or pattern subsection, if there is room to move in the desired direction. It doesn't matter if the item is selected or not.
7. **Section Movement**. If, instead of grabbing the section length handle, one grabs inside the pattern or pattern subsection, that item can be moved horizontally, as long as their is room. Or course, left-clicking inside the item will also cause it to show as selected. One can also highlight a pattern section (making it gray), then click the `p` key to enter "paint" mode, and move the pattern left or right with the arrow keys.
8. **Expansion**. If, instead of grabbing the section length handle, one grabs inside 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.

A feature not yet noted is the ability to split a pattern section in the song editor, either in half or at the nearest snap point to the mouse pointer. The kind of split that is done is determined by the setting of the **File / Options / Mouse / Sequencer64 Options / Middle click splits song trigger at nearest snap (instead of halfway point)** setting. To split a pattern, left-click it to highlight it, move the mouse (if not splitting in half) to the desired pointer, and press `Ctrl-left` on the mouse.

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 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 draw mode, and hold the button. **New:** Just like the Patterns Panel, there is a feature to allow the Mod4 (the Super or Windows) key to keep the right-click in force even after it is released. See 2.2.5.5. Basically, pressing Mod4 before releasing the right-click that allows pattern-adding, keeps pattern-adding in force after the right-click is released. Now pattern can be entered at will with the left mouse button. Right-click again to leave the pattern-adding mode.

New: Another way to turn on the paint mode has been added. To turn on the paint mode, first make sure that the piano roll has the keyboard focus by left-clicking in it, then press the `p` key while in the performance editor. This is just like pressing the right mouse button, but the draw/paint mode sticks (as if the Mod4 mode were in force). While in the paint mode, one can add pattern clips with the left mouse button, via click or drag, and one can highlight a pattern clip and move it with the left and right arrow keys. To get out of the paint mode, press the `x` key while in the sequence editor, to "x-scape" (get it? get it?) from the paint mode. These keys, however, do not work (currently) while the sequence is playing.

New: *Sequencer64*, as of version 0.9.10, now supports zoom in the song editor's piano roll. This feature is not accessible via a button or a menu entry – it is accessible only via keystrokes. After one has left-clicked in the piano roll, the `z`, `Z`, and `0` can be used to zoom the piano-roll view. The `z` key zooms out, the `Z` key zooms in, and the `0` key resets the zoom to the default value. The zoom feature also modifies the time-line (measures indicator).

Then simultaneously left-click the mouse to insert one copy of the pattern. The inserted pattern will show up as a box with a tiny representation of the notes visible inside. (Some patterns, however, 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 on a pattern to drop a new selection position into the pattern, which breaks 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.

When a pattern or a pattern subsection is left-clicked in the piano roll, it is marked with a dark gray filling. When a right-left-hold-drag action is done in this gray area, the result is to *delete* that pattern section or subsection. One can also hit the Delete key to *delete* that pattern section or subsection.

5.2.3 Song Editor / Arrangement Panel / Measures Ruler

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.

Once these anchors 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. One can change the start position only when the performance is not playing.

New: Another way to move the L and R markers, a so-called "movement mode" has been added. To turn on the movement mode, first make sure that the piano roll (not the bar indicator!) has the keyboard focus by left-clicking in it at a blank spot, then press the `l` key or `r` key or while in the sequence editor. *There is no visual feedback that one is in the movement mode.* Then press the left or right arrow key to move the "L" or "R" (depending whether `l` or `r` was used to enter the movement mode) markers by one snap value at a time.

To get out of the movement mode, press the `x` key while in the performance editor, to "x-scape" from the movement mode.

5.3 Song Editor / Bottom Panel

The bottom panel is simple, consisting of a stock horizontal scroll bar and a small button, called the **Grow** button.

The **Grow** button adds to the number of measures that exist in the song editor. The visual effect is very subtle, resulting only in a small change in the thumb of the horizontal scroll-bar, unless one is at the right end of the piano roll. Then, one can see the added measures. Usually about 128 at a time are added, but this depends on the value of PPQN in force.

6 Event Editor

The *Sequencer64 Event Editor* is used to view and edit, in detail, the events present in a sequence/pattern/track.

Warning: This dialog is a new feature of *Sequencer64*, and is still a work-in-progress. Basic viewing and scrolling generally work well, and editing, deleting, and inserting events does work. But there are many possible interactions between event links (Note Off events linked to Note On events), 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 the programmer know!

Also note that this editor is not very sophisticated:

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.
4. It does not have any undo function.
5. It cannot mark more than one event for deletion or modification.
6. There is no support for dragging and dropping of events.

If, some day, we find ourselves needing that kind of functionality, then we can add it. There may also be issues with interactions between the event editor and things like the performance editor and triggers. For now, the event editor is a good way to see the events in a sequence, and to delete or modify problematic events. Unlike the event pane in the pattern editor, this dialog shows all types of events at once.

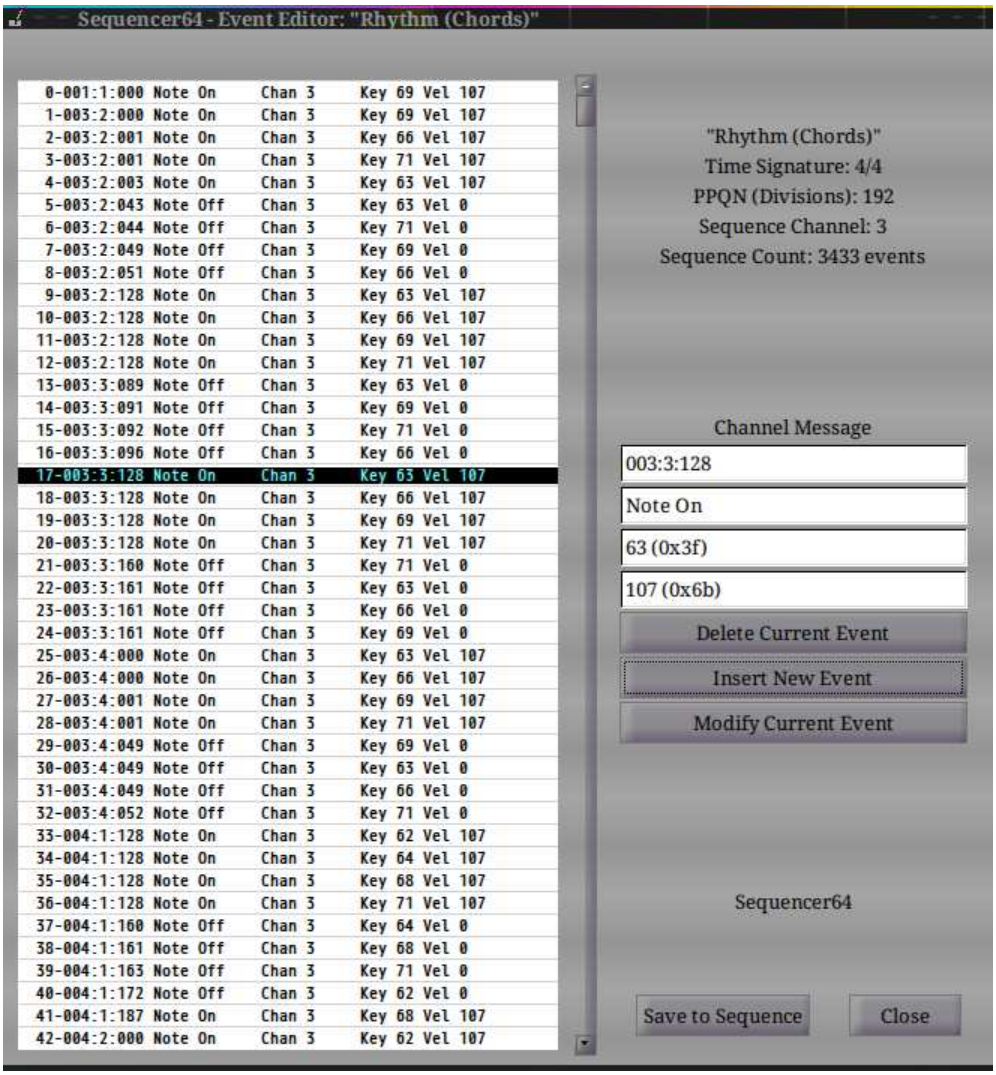


Figure 73: Event Editor Window

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 consists of a list of events, which can be viewed, traversed, and edited. The info fields in the info panel show the name of the sequence containing the events, and some other information about the sequence. The edit fields provide four 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.

Also note that 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.

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   Chan 3    Key 66 Vel 107
```

This line consists of the following parts:

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

1. Index Number. Displays the index number of the event. This number is purely for the reference of the user, and is not part of the event. Events in the pattern are numbered from 0 to the number of events in the pattern. They serve as a way to better know where one is in the sequence.

2. Time Stamp. Displays the time stamp of the event. This value 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.

3. Event Name. Displays the name of the event. 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**

Note that these are all MIDI *channel events*. Support for MIDI *system events* is in place, but is not ready for exposure to the user.

4. Channel Number. Shows the channel number (for channel-events only). Be sure to note that, for the user, MIDI channels always range from 1 to 16. (Internally, they range from 0 to 15).

5. Data Bytes. Shows the one or two data bytes for the event.

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.

6.1.2 Event Frame / Navigation

Moving about in the event frame is fairly straightforward, but has some wrinkles to note. (It was more difficult to get working than expected!)

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" (discussed in the next section). There is currently no support for dragging and dropping events in the event frame.

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, and the updates become very erratic and annoying. 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 sequence being edited and some characteristics of the sequence and the whole song. Currently, five items are shown:

1. **Sequence Name**. This item is redundant, as the window caption for the event editor also shows the sequence name. It can be set in the pattern editor.
2. **Time Signature**. This item is a sequence property. It can be set in the pattern editor.
3. **PPQN** This item shows the "parts per quarter note", or resolution of the whole song. The default PPQN of *Sequencer64* is 192.
4. **Sequence Channel** In *Sequencer64*, the channel number is a property of the sequence. All channel events in the sequence 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 sequence. 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** (read-only)
2. **Event Timestamp**
3. **Event Name**
4. **Data Byte 1**
5. **Data Byte 2**
6. **Delete Current Event**
7. **Insert New Event**
8. **Modify Current Event**

It is important to note that 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. Currently, only channel events can be handled, but someday we hope to handle the wide array of system events, and perhap even system-exclusive events.

2. Event Timestamp. 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 Current Event** 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 Name. 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**

Typing in one of these names should change the kind of event if the event is modified. Abbreviations and case-insensitivity can be used to reduce the effort of typing.

Bug: Currently, the handling of the editing of the event name is a bit clumsy. Also, it would be better to provide a drop-down list for more painless selection of events.

4. Data Byte 1. Allows the 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.

5. Data Byte 2. Allows the 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 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.

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

Sequencer64 would support using the Delete and Insert keys to supplement the buttons, but the Delete key is needed for editing the event data fields. The current structure of the dialog prevents using it for both the frame and the edit fields. Therefore, *Sequencer64* allows the usage of the asterisk keys (both regular and keypad) for deletion.

7. Insert New Event. 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.

8. Modify Current Event. Deletes the current event, and inserts a new event. The modified event is placed in the appropriate location for the given timestamp.

6.4 Event Editor / Bottom Buttons

The buttons at the bottom of the event editor round out the functionality of this dialog.

1. **Save to Sequence**
2. **Close**

1. Save to sequence. Saves the current state of the event container back to the sequence from whence the events came. 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 still exist.

Note that there may still be some subtle bugs in the dialog editor, so be careful about pressing this button.

Also note that any sequence/pattern editor that is open should be reflected in the pattern editor once this button is pressed. However, at present, simultaneous use of the pattern editor and event editor has been disabled.

If both the event editor and the pattern editor are open for a sequence (currently disabled), and some events are deleted in the event editor, and the **Save to Sequence** button is pressed, the pattern editor would crash and takes down *Sequencer64* with it. Therefore, when either editor is open for a given sequence, the right-click menu entries that bring them up are hidden.

2. Close. Closes the event editor. Any unsaved event changes are discarded. There is a "modification indicator" to show that the events have been modified.

Again, good luck with the dialog. Bug reports are appreciated.

7 Sequencer64 Keyboard and Mouse Actions

This section presents some tables summarizing the keyboard and mouse actions available in *Seqeuencer64*. It does not cover the mute keys and the groups keys, which are well described in the keyboard setup for the main window (where "live" performance is controlled; see section 2.2.5.3 "Menu / File / Options / Keyboard" on page 24). It also does not cover the "fruity" mouse actions. Any volunteers to fill in those tables?

This section describes the keystrokes that are currently hardwired in *Sequencer64*. This description only includes items not defined in the **File / Options** dialog. That is, hardwired values. Note that "KP" stands for "keypad".

7.1 Main Window

The main window keystrokes are all defined via the options dialog and "rc" configuration file, or are stock Gtk 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 90. There may be some other keystrokes to be documented at some point.

Table 1: Main Window Support

Action	**Normal**	**Double**	**Shift**	**Ctrl**	**Mod4**
e key	—	—	—	Open song editor	—
Left-click slot	Mute/Unmute	New/Edit	Toggle other slots	Edit	—
Right-click slot	Edit menu	—	Edit menu	Edit/Edit Menu	—

The new mouse features of this window for *Sequencer64*, as noted in section 3 "Patterns Panel" on page 38, 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. It acts like Ctrl-left-click.

7.2 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.

7.2.1 Performance Editor Piano Roll

- `Ctrl-x`. Cut.
- `Ctrl-c`. Copy.
- `Ctrl-v`. Paste.
- `Ctrl-z`. Undo.
- `Ctrl-r`. Redo.
- `Shift-Up`. Move backward one small unit (which is...?)
- `Shift-Down`. Move forward one small unit (which is...?)
- `Shift-Page Up`. Move backward one frame.

- `Shift-Page Down`. Move forward one frame.
- `Shift-Home, Shift-KP Home`. Move to beginning of piano roll.
- `Shift-End, Shift-KP End`. Move to end of piano roll.
- `Shift-z (Z)`. Zoom in.
- `0`. Set default zoom.
- `z`. Zoom out.
- `Left`. Move item left one snap unit.
- `Right`. Move item right one snap unit.
- `Up`. Move frame up one small scroll unit.
- `Down`. Move frame down one small scroll unit.
- `Home`. Move to top of piano roll.
- `End`. Move to bottom of piano roll.
- `Page Up`. Move up one frame (page-increment).
- `Page Down`. Move down one frame (page-increment).

Note that the keystrokes in this table (see table 2 "Performance Window Piano Roll" on page 91) 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**	**Mod4**
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	—	Toggle other slots	—	—
Right-click	Paint mode	—	Paint mode	Paint mode	Lock 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.

Note that 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.

Note that 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. The new features of this window for *Sequencer64*, as noted in section 5 "Song Editor" on page 76, are:

- *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. Please note that opening a second performance view is another way to see more of the performance.
- *0*: Resets the performance view zoom to its normal value.
- *Z*: Zooms in the performance view. It makes the view look bigger, so that more details of the performance can be seen.
- *.*: The period (configurable) is a new key devoted to the new pause functionality.
- *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.
- *Mod4-right-click, release*: Locks the paint mode, until right-click is pressed again later.
- Once selected (and thus 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 – the right mouse button deselects the greyed pattern. Too tricky, we might try fixing it later.

7.2.2 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", and is discussed in section 5.2.3 "Song Editor / Arrangement Panel / Measures Ruler" on page 84. See table 3 "Performance Editor Time Section" on page 92.

Table 3: Performance Editor Time Section

Action	**Normal**	**Double**	**Shift**	**Ctrl**	**Mod4**
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 new features of this window for *Sequencer64*, as noted in section 5.2.3 "Song Editor / Arrangement Panel / Measures Ruler" on page 84, 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.

7.2.3 Performance Editor Names Section

Table 4: Performance Editor Names Section

Action	Normal	Double	Shift	Ctrl	Mod4
Left-Click	Toggle track	—	Toggle other tracks	—	—
Middle-Click	—	—	—	—	—
Right-Click	New/Edit menu	—	—	—	—

7.3 Pattern Editor

The pattern/sequencer editor piano roll is a complex and powerful event editor; table 5 "Pattern Editor Piano Roll" on page 94, doesn't begin to cover its functionality. Here are the keystrokes handle by the main frame of the window:

- `Ctrl-L`. Bring up the LFO event modulation editor, if built into this version (0.9.20 and above) of *Sequencer64*.
- `Ctrl-W`. Exit the sequence (pattern) editor.
- `Ctrl-Page Up`. Zoom in.
- `Ctrl-Page Down`. Zoom out.
- `Shift-Page Up`. Scroll leftward.
- `Shift-Page Down`. Scroll rightward.
- `Shift-Home`. Scroll leftward to the beginning.
- `Shift-End`. Scroll rightward to the end.
- `Shift-z (Z)`. Zoom in.
- `0`. Set default zoom.
- `z`. Zoom out.
- `Page Down`. Scroll downward.
- `Page Up`. Scroll upward.
- `Home`. Scroll upward to the beginning.
- `End`. Scroll downward to the end.

Where is Delete?

7.3.1 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-x`. Cut.
- `Ctrl-c`. Copy.
- `Ctrl-v`. Paste.
- `Ctrl-z`. Undo.
- `Ctrl-r`. Redo.
- `Ctrl-a`. Select all.
- `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!)
- `Left`. Move selected notes one snap left.
- `Down`. Move selected notes one pitch downward.
- `Up`. Move selected notes one pitch upward.
- `Enter, Return`. Paste the selected notes at the current position.
- `p`. Enter "paint" (also known as "adding") mode.
- `x`. Escape ("x-scape") the paint mode.

And here is the table:

Table 5: Pattern Editor Piano Roll

Action	**Normal**	**Double**	**Shift**	**Ctrl**	**Mod4**
Del	Delete Selected	—	—	—	—
c	—	—	—	Copy	—
p	Paint mode	—	—	—	—
v	—	—	—	Paste	—
x	Escape Paint	—	—	Cut	—
z	Zoom Out	—	—	Undo	—
0	Reset Zoom	—	—	—	—
Z	Zoom In	—	—	Undo?	—
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	Lock Paint mode
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	—

- 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.

The new features of this window section for *Sequencer64*, as noted in section 4.3.1 "Pattern Editor / Piano Roll Items" on page 66, are:

- *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. Please note that opening a second performance view is another way to see more of the performance.
- *0*: Resets the performance view zoom to its normal value.
- *Z*: Zooms in the performance view. It makes the view look bigger, so that more details of the performance can be seen.
- *.*: The period (configurable) is a new key devoted to the new pause functionality.
- *Left Arrow*: Moves the selected events to the left (earlier in time) in the performance layout.
- *Right Arrow*: Moves the selected events to the right (later in time) in the performance layout.
- *Up Arrow*: Moves the selected notes upward in direction and pitch.
- *Down Arrow*: Moves the selected notes downward in direction and pitch.
- *Mod4-Right-Click*: Locks the paint mode, until right-click is pressed again later.

7.3.2 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.

7.3.3 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.

7.3.4 Pattern Editor Virtual Keyboard

Table 6: Pattern Editor Virtual Keyboard

Action	**Normal**	**Double**	**Shift**	**Ctrl**	**Mod4**
Left-Click	Play note	—	—	—	—
Right-Click	Toggle labels	—	—	—	—

7.4 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.

8 Sequencer64 "rc" Configuration File

The *Sequencer64* configuration file originally was `.seq24rc`, and it was stored in the user's `$HOME` directory. This is the same name used by *Seq24*, so we created an new file to take its place, with a fall-back to the original file-name if the new file does not exist, or if *Sequencer64* is running in legacy mode.

After you run *Sequencer64* for the first time (in non-legacy mode), it will generate a `sequencer64.rc` file in your home directory:

```
/home/ahlstrom/.config/sequencer64/sequencer64.rc
```

It contains the the data for remote MIDI control, keyboard control, MIDI clock, and a few other settings. (See section 9 "Sequencer64 "usr" Configuration File" on page 113for some more settings.)

Sequencer64 will overwrite the `sequencer6.4rc` file upon quitting. One should therefore quit *Sequencer64* before doing manual modifications to the `sequencer64.rc` file.

Note that there is an old, but complete, example of the *Seq24* "rc" file at [20]. It includes a setup for the Novation Launchpad device.

8.1 Sequencer64 "rc" File / MIDI Control Section

Like *Seq24*, *Sequencer64* provides a way to control the application to some extent via a MIDI controller, such as a MIDI keyboard or a MIDI pad device. The current section describes this feature; additional resources and ideas can be found at `linuxaudio.org` ([10]).

For each pattern, we can set up MIDI events to turn a pattern on, off, or to toggle it. This setup is in the MIDI Control section of `sequencer64.rc`, and begins with an "INI-style" group marker `[midi-control]`.

Each MIDI control line has the following format:

```
74  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
```

The leftmost brackets define a *toggle* filter; the middle brackets define a *on* filter; the rightmost brackets define a *off* filter. The numbers inside the brackets define six values that set up the control: **on/off**; **inverse**; **MIDI status byte**; **data1**; **data2 min**; and **data2 max** This layout of values is explained in more detail below.

The MIDI control setup resembles a matrix. The first block of matrix elements represents control for control functions of the active screen-set. These entries are numbered from 0 to 63. Legacy control keys occupy entries from 64 to 73. We've added some more entries, from 74 to 83, to control additional *Sequencer64* functions.

The MIDI Control section is explicitly broken into subsections, though those subsections are marked with comment-lines for better comprehensibility. The subsections of the MIDI Control section are:

1. **Pattern group**. Consists of 32 lines, one for each pattern box shown in the Pattern window. It provides a way to control the arming/disarming (muting/unmuting) of each pattern shown in the main window. Note that the main window shows the *active* screen-set. These controls affect the *active* screen-set.

2. **Mute-in group**. Consists of 32 lines, one for each pattern box shown in the Pattern window. 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.
3. **Automation group**. Each item in this group consists of one line. Each line specifies a MIDI event that can cause the given operation to occur. Now, from our vantage point, the "up" versus "down" functions should not need two entries to effect them... we should be able to use the "on" control to perform "up", and the "off" control to perform "down". But this is the legacy setup and we shall respect that.
    1. **bpm up**. This MIDI control increments the beats-per-minute setting, as if the up-arrow has been clicked, or the up-arrow key pressed, in this control. 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. See section 9.4 "Sequencer64 "usr" File / User MIDI Settings" on page 122.
    2. **bpm down**. Similarly, this MIDI control decrements the beats-per-minute setting, as if the down-arrow has been clicked/pressed.
    3. **screen-set up**. This MIDI control increments to the next screen-set. Once the screen-set has been altered, mute-groups and other actions apply to that screen set.
    4. **screen-set down**. Similarly, this MIDI control decrements to the previous screen-set.
    5. **mod replace**. This MIDI control sets the "replace" status flag. Then, when the user manually clicks a pattern slot, that pattern is unmuted, and all the rest are muted. This works whether in "Live" or "Song" mode.
    6. **mod snapshot**. This MIDI control causes the playing status 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. Compare to section section 3.2.3.1 "Pattern Keys" on page 50for a better idea of how it works.
    7. **mod queue**. This MIDI 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.
    8. **mod gmute**. This MIDI control sets up a "mute group". More to come on this one.
    9. **mod glearn**. This MIDI control sets up a "group learn". However, as the group-learn key is a modifier key that needs to be held, we're not quite sure how this works with MIDI control.
    10. **screen-set play**. This MIDI control sets the playing screen-set, but we're not quite sure how it works yet.
4. **Extended automation group**. These additional control items were requested by users, to control addition features of the application. Each item in this group consists of one line.
    1. **stop/pause/start**. Emulate the Stop, Pause, and Start keys, using Toggle for pause, Off for stop, and On for start.
    2. **record**. For recording a live performance by recording the mute/unmute states that the musician played. Not yet functional.
    3. **solo on/off**. Not yet functional.
    4. **thru toggle**. Not yet functional.
    5. **reserved for expansion** (six of these are reserved).

For all of these MIDI control lines, the three fields, each between the brackets, on each line, correspond to a *MIDI filter* to toggle, enable, or disable a sequence, change a selection, or activate a feature. If the incoming MIDI event value matches a value present in the filter, it will *toggle* (first field), *enable* (second field) or *disable* (third field) the sequence.

We see the following lines in the MIDI Control section, which is broken into groups or subsections marked by comments:

```
[midi-control]
74      # MIDI controls count

# pattern group
 0  [0 0 0 0 0 0]   [1 0 144  96 0 127]   [1 0 128  96 0 127]
 1  [0 0 0 0 0 0]   [1 0 144  97 0 127]   [1 0 128  97 0 127]
 2  [0 0 0 0 0 0]   [1 0 144  98 0 127]   [1 0 128  98 0 127]
...     ...             ...                ...
31  [0 0 0 0 0 0]   [1 0 144 127 0 127]   [1 0 128 127 0 127]

# mute in group section:
32  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
33  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
...     ...             ...                ...
63  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]

# bpm up:
64  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# bpm down:
65  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# screen set up:
66  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# screen set down:
67  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# mod replace:
68  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# mod snapshot:
69  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# mod queue:
70  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# mod gmute:
71  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# mod glearn:
72  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# screen set play:
73  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
```

The number (74) is the number of lines in the MIDI Control section. The new extended automation values bring this number up to 84:

```
# Extended MIDI controls:
# start playback (pause, start, stop):
74  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# performance record:
75  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# solo (toggle, on, off):
76  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
```

```
# MIDI THRU (toggle, on, off):
77  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# bpm page up:
78  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# bpm page down:
79  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# reserved for expansion:
80  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# reserved for expansion:
81  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# reserved for expansion:
82  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
# reserved for expansion:
83  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
```

Not all of these new values are yet usable. Currently, just the "start", "pause", "stop", and "bpm" page controls have been implemented.

So let's concentrate on one line of data:

```
74  [0 0 0 0 0 0]   [0 0 0 0 0 0]   [0 0 0 0 0 0]
```

The first number represents one of the following entities, depending on its value:

- **0 to 31**. These items represent the **Pattern group** for patterns (sequences) 1 to 32.
- **32 to 63**. These items represent the **Mute-in group** for patterns (sequences) 1 to 32.
- **64 to 73**. These items represent the legacy *Seq24* controls, from **bpm up** to **screen set play**.
- **74 to 83**. These items, if present, represent the new extended controls, to control addition playback features of *Sequencer64*.

Each set of brackets on the line corresponds to a "MIDI filter":

- The leftmost bracket set defines the *toggle* filter.
- The middle bracket set defines the *on* filter.
- The rightmost bracket set defines the *off* filter.

If the incoming MIDI event matches the filter, it will either [**toggle**], [**on**], or [**off**] the pattern/sequence, respectively. The layout of each filter inside the brackets is as follows:

[**OPR INV STAT D1 D2min D2max**]

- **OPR = on/off**
- **INV = inverse**
- **STAT = MIDI status byte** (channel ignored)
- **D1 = data1**
- **D2min = data2 min**

- **D2max = data2 max**

If **OPR (on/off)** is set to 1, it will match the incoming MIDI against the **STAT (MIDI status byte)** pattern. and perform the action (on/off/toggle) if the data falls in the range specified. All values are in decimal.

Note: In legacy versions (*Seq24* and early versions of *Sequencer64*), the channel nybble of the MIDI control (and all other incoming MIDI events) were stripped off. This is no longer the case, and thus opens up many more events useful for MIDI control. But do note that events that are actually recorded end up getting the channel number of the pattern into which they are recorded.

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 decimals. The channel nybble of this byte is ignored. One can look the possible status values up in the MIDI messages tables; the relevant data can be found at [11]. As the channel on which the events are sent is ignored, it is sufficient to use the values for channel 1. That is, 0.

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.

8.1.1 Sequencer64 "rc" File / MIDI Control Pattern Group

Complex? Here is an example for the some of the first 32 lines, which comprise the *pattern group*. The following is an example of responding to Note On events for note 0, with any velocity, to turn the pattern on, and Note Off events for note 0, and any velocity, to turn the pattern off.

```
      Toggle                  On                      Off
1 [0 0 0 0 0 0]       [1 0  144 0 0 127]       [1 0 128 0 0 127]
```

The first number, 1, indicates the second pattern (pattern numbering starts from 0). The first section, **Toggle**, is off (inactive). All values are 0. There is no setup to use MIDI control to toggle pattern 1 here.

On to the second section, **On**:

- The **On** section starts with **OPR** = 1, so it is on (1 = active).
- The **inverse** value is off (0 = inactive).
- The **MIDI status byte**, 144, which is 0x90 (hex), which is a Note On event on channel 0. However, the channel is ignored.
- The **data1** values sets the actual Note value to 0, meaning the lowest possible MIDI note (pitch) value.
- **data2 min** value sets the minimum value to 0.

- **data2 max** sets the maximum value to 127.

Thus, receiving any Note On velocity for note 0 will turn sequence 1 *on*. This is the second pattern; in the default setup, key `q` would operate on this pattern as well.

On to the **Off** section:

- The **Off** field is on (active).
- The **inverse** value is off (0 = inactive).
- The **MIDI status byte**, 128, which is 0x80 (hex), which 128, which is 0x80 (hex), which is a Note Off event on channel 0.
- The **data1** values sets the actual Note value to 0, meaning the lowest possible MIDI note (pitch) value.
- **data2 min** value sets the minimum value to 0.
- **data2 max** sets the maximum value to 127.

Thus, receiving any Note Off velocity for note 0 will turn sequence 1 *off*.

So, basically, pattern 1 starts when any Note On for MIDI note 0 is received, and it stops when any Note Off for MIDI note 0 is received. One can easily extend this so that Note On/Off values from 0 to 31 control the corresponding pattern slot.

Obviously, one might not want Note On/Off events from any channel to trigger events, so some other event would likely be more useful. (Hmmmm, we could add an option to not strip the channel value....)

The following example would map a row of sequences to one knob sending out changes for Control Code 1:

```
      Toggle                  On                          Off
0 [0 0 0 0 0 0]       [1 1 176 1   0   15]      [0 0 0 0 0 0]
1 [0 0 0 0 0 0]       [1 1 176 1  16   31]      [0 0 0 0 0 0]
2 [0 0 0 0 0 0]       [1 1 176 1  32   47]      [0 0 0 0 0 0]
3 [0 0 0 0 0 0]       [1 1 176 1  48   63]      [0 0 0 0 0 0]
4 [0 0 0 0 0 0]       [1 1 176 1  64   79]      [0 0 0 0 0 0]
5 [0 0 0 0 0 0]       [1 1 176 1  80   95]      [0 0 0 0 0 0]
6 [0 0 0 0 0 0]       [1 1 176 1  96  111]      [0 0 0 0 0 0]
7 [0 0 0 0 0 0]       [1 1 176 1 112  127]      [0 0 0 0 0 0]
```

The **on** field is on (active). Inverse is active. The **MIDI status byte**, 176, is 0xB0 (hex), which is a Control Change event (channel ignored). **data1** is 1, which is the controller number for a Modulation Wheel. The **data2** ranges are set so that, as the controller data increases (as the modulation-wheel knob is turned, so to speak), patterns 0 through 7 come on one at a time until all are running.

Here is another example from [10], 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:

```
# mod queue
#  [   toggle-filter       ] [      on-filter       ] [      off-filter   ]

70 [0   0   0   0   0   0  ] [1   0   176 64 127 127] [1   0  176 64  0  0]

#   OPR INV STA D1  mn mx     OPR INV STA D1 mn  mx   OPR 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. Here's a little table of the decimal numbers for some commonly-used MIDI controls:

- **128** or **129** for any Note On or Note Off events.
- **160** Polyphonic aftertouch.
- **176** Controller event.
- **192** Program change.
- **208** Aftertouch.
- **224** Pitch wheel.

8.1.2 Sequencer64 "rc" File / MIDI Control Mute In Group

This section controls 32 groups of mutes in the same way as defined for `[midi-control]`, and is in fact placed in the `[midi-control]` section. 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.

What is the different between the **mute-in group** section and the **mute group** section? The former defines the MIDI control values that can affect the muting of a group, while the latter specifies the armed patterns that are part of a group.

8.1.3 Sequencer64 "rc" File / MIDI Control Automation Group

1. **bpm up.** Increases the BPM (speed) of the sequencer based on MIDI input.
2. **bpm down.** Decreases the BPM (speed) of the sequencer based on MIDI input.
3. **screen-set up.** Increases the active screen-set of the sequencer based on MIDI input.
4. **screen-set down.** Decreases the active screen-set of the sequencer based on MIDI input.
5. **mod replace.** This item provides a way to automate replacement.
6. **mod snapshot.** This item provides a way to automate snapshots.
7. **mod queue.** This item provides a way to automate queueing.
8. **mod gmute.** This item provides a way to automate group-muting.
9. **mod glearn.** This item provides a way to automate group-learning.
10. **screen-set play.** This item provides a way to automate screen set play.

8.2 Sequencer64 "rc" File / MIDI Control Extended Automation Section

This section shows how to set up an extended automation control.

Currently, this control is enabled only in the **wip** branch of the *Sequencer64* project, and the only extended control that works is the *stop/pause/start* functionality (which requires that the pause functionality be enabled at build time, which is the default).

Here, we will set up *Sequencer64* so that the first three MIDI white keys (notes 0, 2, and 4) will become "Stop", "Pause", and "Start" buttons.

The first step, if not already done, is to install the new version (**0.90.2 * wip** and above) of *Sequencer64*, run it, and then exit. Verify in the regenerated "rc" file (`~/.config/sequencer64/sequencer64.rc`) that the following lines exist:

```
84      # MIDI controls count (74 or 84)
```

and

```
# Extended MIDI controls:
# start playback (pause, start, stop):
74 [0 0   0   0   0   0] [0 0   0   0   0   0] [0 0   0   0   0   0]
```

MIDI control 74's toggle/on/off sections will be used to implement the stop/pause/start functionality. Replace the "74" line with the following line:

```
74 [1 0 144   2   0 127] [1 0 144   4   0 127] [1 0 144   0   0 127]
```

This sets up MIDI Note On (144) values 2 (toggle/pause), 4 (on/start), and 0 (off/stop). Now we are ready to test this feature. One can use a MIDI keyboard to do so, but here we will use the *VMPK* ([29]) virtual MIDI piano keyboard application for this test. Refer to the following figure.

In addition, let's set up the standard and the new BPM (beats/minute) MIDI control values.

```
# bpm up: Note On 9
64 [0 0   0   0   0   0] [1 0 144   9   0 127] [0 0   0   0   0   0]
# bpm down: Note On 7
65 [0 0   0   0   0   0] [1 0 144   7   0 127] [0 0   0   0   0   0]
```

That section sets up the standard (step-size) BPM controls, which correspond to the fine control possible with the up and down arrows of the BPM spinner in the main window. It sets up Note On 7 to be the BPM-down control, and Note On 9 to be the BPM-up control. These two keys are the dark-cyan keys shown in the figure.

```
# bpm page up:  Note On 11
78 [0 0   0   0   0   0] [1 0 144  11   0 127] [0 0   0   0   0   0]
# bpm page down:  Note On 5
79 [0 0   0   0   0   0] [1 0 144  05   0 127] [0 0   0   0   0   0]
```

That section uses the extended controls (74 to 83) set up the coarse (page-size) BPM controls, which correspond to the larger jumps possible with the Page Up and Page Down keys when the BPM spinner has focus. It sets up Note On 5 to be the BPM-page-down control, and Note On 11 to be the BPM-page-up control. These two keys are the bright-cyan keys shown in the figure.

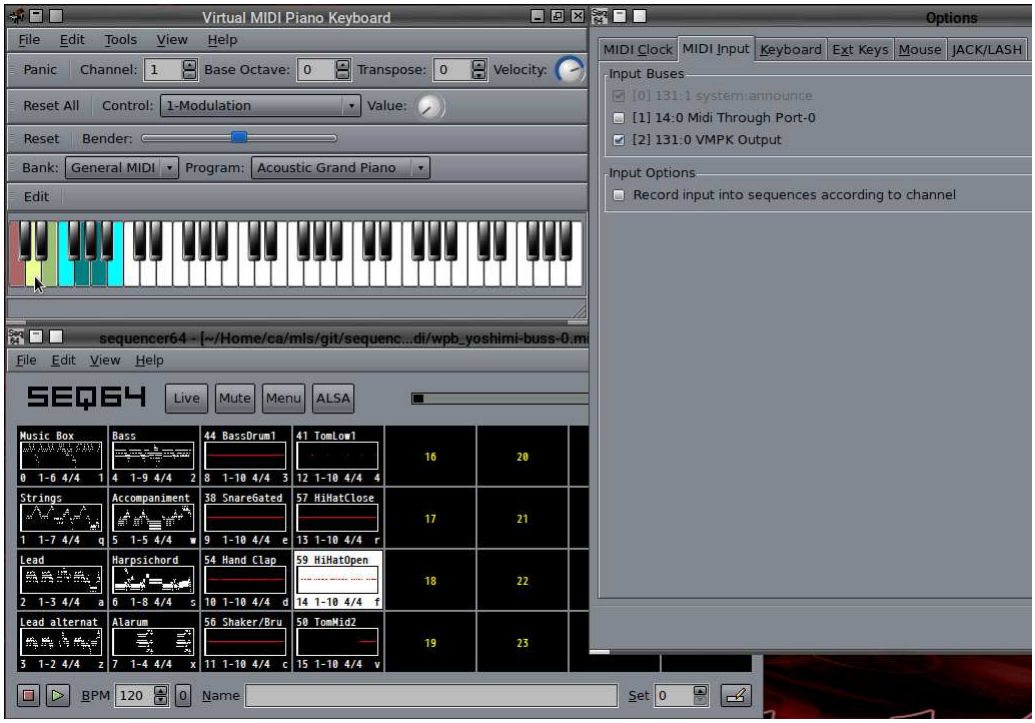


Figure 74: Stop/Pause/Start ALSA Test Setup

One can copy these settings from the sample file `contrib/simple-midi-control-section.rc`, if one wants to try them.

Set up *VMPK* to use the lowest octave by setting **Base Octave** to `0`. The red, yellow, and green keys shown will be our stop, pause, and start keys.

Next, run *Sequencer64*, using the following command line to make sure that it is using ALSA and using automatic mode to connect the ALSA MID ports:

```
$ seq64 -A -a
```

Then open a MIDI file. Next, open the **File / Options / MIDI Input** tab, and make sure that the **VMPK Output** is check-marked as shown in the figure. If desired, also connect up to some kind of synthesizer so that the song can be heard.

Finally, press the third white key (shown as green in the figure) to start playback. The second white key (yellow in the figure) will pause and resume playback. The first white key (red in the figure) will stop (and rewind) playback.

Then play with the BPM MIDI control keys. Note that the size of the BPM step-increment and the BPM page-increment are configurable in the `[user-midi-settings]` section of the "usr" configuration file, using the following values in that section:

```
1        # bpm_precision
1.0      # bpm_step_increment
10       # bpm_page_increment
```

See section 9.4 "Sequencer64 "usr" File / User MIDI Settings" on page 122; it has information about the usage and enabling of these settings.

Obviously, this setup is not useful for performance, but serves as a good example to verify this MIDI control.

One thing we noticed while implementing this functionality is that there is really no need to have two lines for pairs such as BPM up/down and screen-set up/down. Also, is screen-set play now partly redundant? No matter, we will not break the user's existing setup.

8.3 Sequencer64 "rc" File / Mute-Group Section

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-group]
1024    # group mute value count
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] [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] [0 0 0 0 0 0 0 0]
2 [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 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] [0 0 0 0 0 0 0 0]
```

The initial number, 1024 is probably the total count of 32 x 32 sequences. In this group are the definitions of the state of the 32 sequences in the playing screen set when a group is selected. Each set of brackets defines a group:

```
[state of the first 8 sequences] [second 8] [third 8] [fourth 8]
```

After the list of sequences and their MIDI events, one can set *Sequencer64* to handle MIDI events and change some more settings in `sequencer64.rc`.

What is the different between the **mute-in group** section and the **mute group** section? The former defines the MIDI control values that can affect the muting of a group, while the latter specifies the patterns that are part of a group.

8.4 Sequencer64 "rc" File / MIDI-Clock Section

The MIDI Clock fields will contain the clocking state from the last time *Sequencer64* was run. Turn off the clock with a 0, or on with a 1. This section has 16 entries, one for each MIDI output buss that *Sequencer64* supports.

This configuration item is the same as the **MIDI Clock** tab described in paragraph 2.2.5.1 "Menu / File / Options / MIDI Clock" on page 20

Here is the format:

```
[midi-clock]
16
 0 0  #  [1] seq24 1
 1 0  #  [2] seq24 2
 2 0  #  [3] seq24 3
 3 0  #  [4] seq24 4
 4 0  #  [5] seq24 5
 5 0  #  [6] seq24 6
 6 0  #  [7] seq24 7
 7 0  #  [8] seq24 8
 8 0  #  [9] seq24 9
 9 0  # [10] seq24 10
10 0  # [11] seq24 11
11 0  # [12] seq24 12
12 0  # [13] seq24 13
13 0  # [14] seq24 14
14 0  # [15] seq24 15
15 0  # [16] seq24 16
```

That sample would be written one had started up *Sequencer64* in manual-alsa-mode. On our system, where we have Timidity running, and erroneously have also specified 3 MIDI busses that we do not have, in the `sequencer64.usr` file:

```
[midi-clock]
5    # number of MIDI clocks/busses
# Output buss name: [0] 14:0 2x2 A (SuperNova,Q,TX81Z,DrumStation)
0 0  # buss number, clock status
# Output buss name: [1] 128:0 2x2 B (WaveStation,ESI-2000,MV4,ES-1,ER-1)
1 0  # buss number, clock status
# Output buss name: [2] 128:1 PCR-30 (303)
2 0  # buss number, clock status
# Output buss name: [3] 128:2 TiMidity port 2
3 0  # buss number, clock status
# Output buss name: [4] 128:3 TiMidity port 3
4 0  # buss number, clock status
```

8.5 Sequencer64 "rc" File / Keyboard Control Section

The keyboard control is a dump of the keys that *Sequencer64* recognises, and each key's corresponding sequence number. Note that the first number corresponds to the number of sequences in the active screen set.

```
[keyboard-control]
32     # number of keys
# Key #  Sequence #   Key name
44  31         # comma
49  0          # 1
50  4          # 2
51  8          # 3
52  12         # 4
53  16         # 5
54  20         # 6
55  24         # 7
56  28         # 8
97  2          # a
98  19         # b
99  11         # c
100  10        # d
101  9         # e
102  14        # f
103  18        # g
104  22        # h
105  29        # i
106  26        # j
107  30        # k
109  27        # m
110  23        # n
113  1         # q
114  13        # r
115  6         # s
116  17        # t
117  25        # u
118  15        # v
119  5         # w
120  7         # x
121  21        # y
122  3         # z
```

8.6 Sequencer64 "rc" File / Keyboard Group Section

This section is the same as [**keyboard-control**], but to control groups of patterns, rather than individual patterns, using keystrokes. The keyboard group specifies more automation for the application. The first number specifies the key number, and the second number specifies the Group number.

Additional control items:

1. **# bpm up and down**. Keys to control BPM (beats per minute).
2. **# screen set up and down**. Keys for changing the active screenset.
3. **# group functionality on, off, learn**. Note that the group learn key is a modifier key to be held while pressing a group toggle key.
4. **#replace, queue, snapshot_1, snapshot_2, keep queue**. These are the other modifier keys explained in section 3a.

To see the required key codes when pressed, run `seq24` with the `--show-keys`.

Some keys should not be assigned to control sequences in *Sequencer64* as they are already assigned in the *Sequencer64* menu (with `Ctrl`).

This configuration item is the same as the **Keyboard** tab described in section 2.2.5.3 "Menu / File / Options / Keyboard" on page 24.

```
[keyboard-group]
# Key #, group #
32
33  0         # exclam
34  1         # quotedbl
35  2         # numbersign
36  3         # dollar
37  4         # percent
38  5         # ampersand
40  7         # parenleft
47  6         # slash
59  31        # semicolon
65  16        # A
66  28        # B
67  26        # C
68  18        # D
69  10        # E
70  19        # F
71  20        # G
72  21        # H
73  15        # I
74  22        # J
75  23        # K
77  30        # M
78  29        # N
81  8         # Q
82  11        # R
83  17        # S
84  12        # T
85  14        # U
86  27        # V
87  9         # W
88  25        # X
```

```
89  13        # Y
90  24        # Z
39 59         # bpm up, down: apostrophe semicolon
93 91 65360   # screen set up, down, play: bracketright bracketleft Home
236 39 65379  # group on, off, learn: igrave apostrophe Insert
# replace, queue, snapshot_1, snapshot 2, keep queue:
65507 65508 65513 65514 92  # Control_L Control_R Alt_L Alt_R backslash
1             # show_ui_sequence_key (1=true/0=false)
32            # space start sequencer
65307         # Escape stop sequencer
0 #  show sequence numbers (1 = true / 0 = false);  ignored in legacy mode
```

Note that most of these group-control keys are shifted versions of the keystrokes that control the individual sequences. Also note the `Control_L` and `Control_R` notations a few lines above. Please avoid using any Control key combinations in the "rc"/Keyboard configuration. Control keys are the province of the user-interface (*Gtk+*) and assigning them can cause surprising behavior! It is also wise to avoid the `Alt` key.

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*.

8.7 Sequencer64 "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, as indicated in the comments below.

This configuration item is the same as the **Jack Sync** tab described in section 2.2.5.6 "Menu / File / Options / Jack Sync and LASH" on page 31.

```
[jack-transport]

# jack_transport - Enable slave sync with JACK Transport.
0

# jack_master - Sequencer64 attempts to serve as JACK Master.
0

# jack_master_cond - Sequencer64 is master if no other master exists.
0

# song_start_mode (applies mainly if JACK is enabled)
# 0 = Playback in live mode. Allows muting and unmuting of loops.
# 1 = Playback uses the song editor's data.
1
```

An additional item, new, specifies if native JACK MIDI input/output is to be used.

```
# jack_midi - Enable JACK MIDI, which is a separate option from
# JACK Transport.
1
```

Please note that only *one* of jack_transport, jack_master, and jack_master_cond should be selected (set to 1) at a time. Also note that JACK transport is separately configurable from JACK MIDI, and each uses a different JACK client internally.

8.8 Sequencer64 "rc" File / Other Sections

This configuration item is the same as the **Clock Start Modulo** option described in paragraph 2.2.5.1 "Menu / File / Options / MIDI Clock" on page 20.

```
[midi-clock-mod-ticks]
64
```

This configuration item is the same as the **MIDI Input** tab described in paragraph 2.2.5.2 "Menu / File / Options / MIDI Input" on page 22. The "1" is undoubtedly a record count, and would equal the number of supported input ports. This "rc" entry here has two variables; the first is the record number or port number, and the second number indicates whether it is disabled (0), or enabled (1).

```
[midi-input]
1   # number of MIDI busses
# [0] 14:0 2x2 A (SuperNova,Q,TX81Z,DrumStation)
0 0
```

There is no user-interface item for the following value, but it does correspond to the `--manual-alsa-ports` command-line option.

```
# set to 1 if you want seq24 to create its own alsa ports and
# not connect to other clients

[manual-alsa-ports]
1
```

The opposite of `--manual-alsa-ports` is `--auto-alsa-ports`. The auto-alsa-ports option forces *Sequencer64* to use the system's existing ALSA ports. This is necessary in order to play tunes through software synthesizers that use ALSA MIDI.

Turning on the manual-alsa-ports option is necessary if one wants to use the legacy *Sequencer64* (`sequencer64`) with JACK. It is *not* necessary if using the native JACK MIDI version, `seq64`. However, if one needs to avoid the auto-connect feature of `seq64`, then the manual option is necessary.

It will create ports as per the settings in the "user" configuration file's `user-midi-bus-definitions` and `user-midi-bus-N` sections. 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. However, this option is misleading if one wants to have access to the actual ALSA ports that exist on the system. The next option gets around that issue.

```
# Set to 1 to have sequencer64 ignore any system port names
# declared in the 'user' configuration file.  Use this option if
# you want to be able to see the port names as detected by ALSA.

[reveal-alsa-ports]
1   # flag for reveal ALSA ports
```

Turning on the reveal-alsa-ports option is necessary if one wants to see the actual ALSA port names defined by the system. It will ignore the settings in the "user" configuration file's `user-midi-bus-definitions` and `user-midi-bus-N` sections. If this option is turned on, the definitions in the "user" configuration file are *not* read from that file.

This configuration item is the same as the **Mouse** tab described in paragraph 2.2.5.5 "Menu / File / Options / Mouse" on page 30.

```
# 0 - 'seq24' (original seq24 method)
# 1 - 'fruity' (similar to a certain fruity sequencer we like)

[interaction-method]

# 0 - 'seq24' (original seq24 method)
# 1 - 'fruity' (similar to a certain fruity sequencer we like)

0   # interaction_method
```

New: There is now an option to use the Mod4 (Super, or Windows) key in the Pattern Editor to lock the editing of a note. When this mode is enabled, and Mod4 is pressed while the mouse right-button is released, the editing pencil icon remains, and notes can be added. This feature is useful for crippled trackpads and trackpad drivers that cannot provide two simultaneous button presses.

```
# Set to 1 to allow seq24 to stay in note-adding mode when
# the right-click is released while holding the Mod4 (Super or
# Windows) key.

1   # allow_mod4_mode
```

New: 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.

```
# Set to 1 to allow Sequencer64 to split performance editor
# triggers at the closest snap position, instead of splitting the
# trigger exactly in its middle.  Remember that the split is
# activated by a middle click.

0   # allow_snap_split
```

New: 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.

```
# Set to 1 to allow a double-click on a slot to bring it up in
# the pattern editor.  This is the default.  Set it to 0 if
# it interferes with muting/unmuting a pattern.

1   # allow_click_edit
```

The following configuration item is the same as the `--lash` or `--no-lash` options described in section 10 "Sequencer64 Man Page" on page 126. If set to 0, LASH session support is disabled. If set to 1, LASH session support is enabled. However, if LASH support is not built into the application, neither option has any effect – there is no LASH support. To determine if LASH support is built in, run sequencer64 from the command line with the `--version` option, and see if LASH is mentioned.

```
[lash-session]
# Set the following value to 0 to disable LASH session management.
# Set the following value to 1 to enable LASH session management.
# This value will have no effect is LASH support is not built into
# the application.  Use the --help option to see if LASH is part of
# the options list.
1     # LASH session management support flag
```

This new item determines if the "rc" configuration file is saved upon exit of *Sequencer64*. The legacy behavior is to save it, which can sometimes be inconvenient when one is just trying out some command-line options.

```
[auto-option-save]
# Set the following value to 0 to disable the automatic saving of the
# current configuration to the 'rc' file.  Set it to 1 to
# follow legacy seq24 behavior of saving the configuration at exit.
# Note that, if auto-save is set, many of the command-line settings,
# such as the JACK/ALSA settings, are then saved to the configuration,
# which can confuse one at first.  Also note that one currently needs
# this option set to 1 to save the configuration, as there is not a
# user-interface control for it at present.
0     # auto-save-options-on-exit support flag
```

The following item refers to the last directory in which one opened or saved a MIDI file.

```
[last-used-dir]

# Last used directory.

/home/ahlstrom/Home/ca/mls/git/sequencer64/contrib/midi/
```

9 Sequencer64 "usr" Configuration File

The *Sequencer64* "usr" ("user") configuration file provides a way to give more informative names to the MIDI busses, MIDI channels, and MIDI controllers of a given system setup. This configuration will override the default values of some drop-down lists and menu items, and make them reflect your names for them. It now also includes some items that affect the user-interface's look, and some new functionality.

Unlike the "rc" file, the "user" file is not written every time *Sequencer64* exits. If the "user" files does not exist, one is created, but it is normally not overwritten thereafter. To cause it to be overwritten at exit, run *Sequencer64* with the `-u` or `--user-save` option:

```
$ seq64 --user-save
```

This option is also useful when one installs a new version of *Sequencer64* that adds new options to the "user" file. See section 10 "Sequencer64 Man Page" on page 126; it discusses more options involving the "user" file.

Another difference between the "rc" file and the "user" file is that the "user" file currently has no graphical user-interface dialog to configure the "user" settings. One has to edit the file manually.

The original purpose for the "user" file was to create familiar names for the system MIDI devices. By default, the list of MIDI devices that *Sequencer64* shows depends on one's system setup and whether the manual-alsa-port option is specified or not. Here's our system, which has Timidity installed and running as a service, and the `[manual-alsa-port]` option turned off, shown in a composite view with all menus one can look at for MIDI settings:

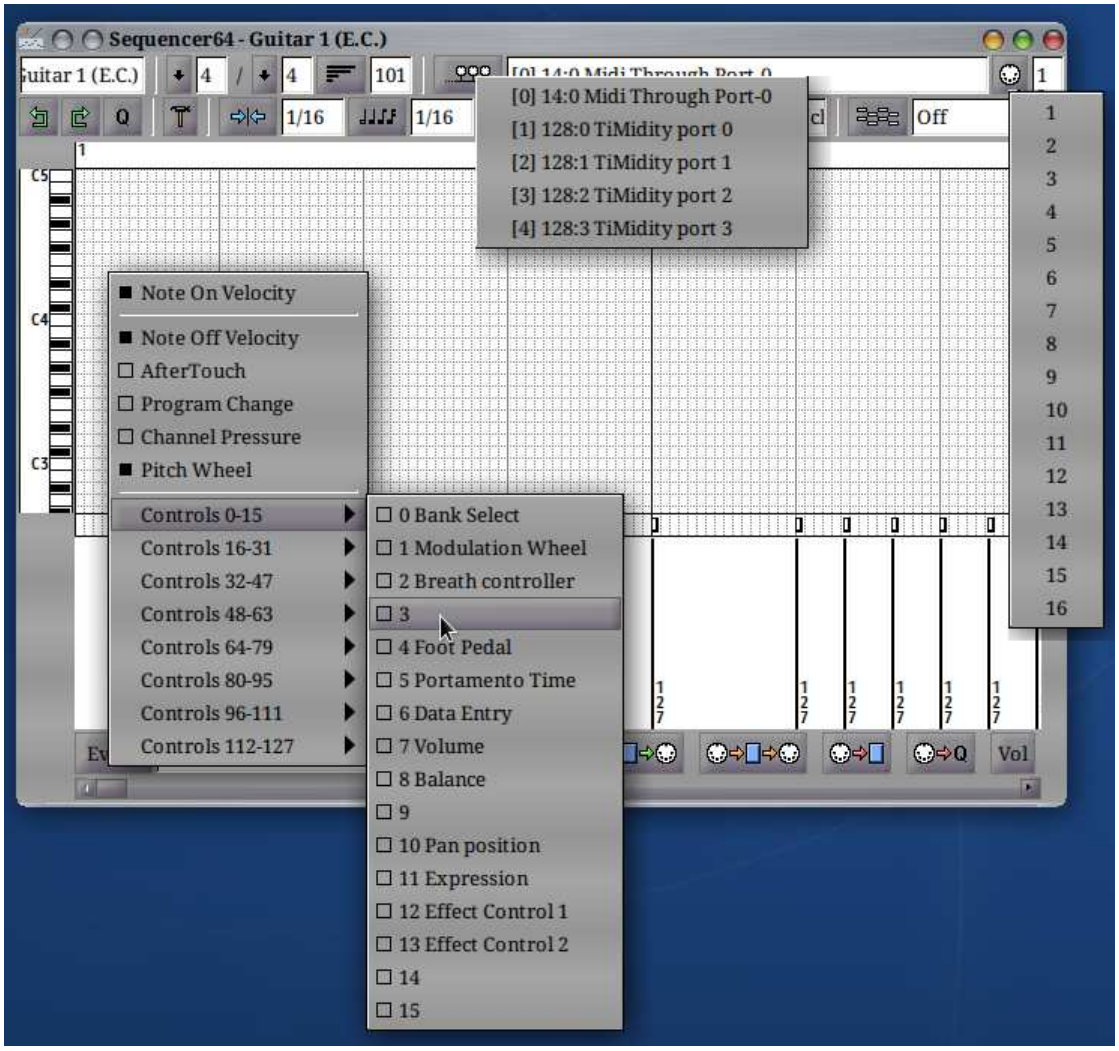


Figure 75: Sequencer64 Composite View of Native Devices

At the top center, the dropdown menu contains the 5 MIDI busses (also known as "MIDI ports") supported by our system. 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 *Sequencer64* 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.

Now let's 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 affecting 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 (apparently) to route certain MIDI channels to certain MIDI devices. Let's assume we have them set up this way:

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

How can we get *Sequencer64* to show these items with the proper names associated with each device, channel, and controller value? We use the oddly-named **"user" configuration file**.

The *Seq24* configuration file was called `.seq24usr`, and it was stored in the user's `$HOME` directory. For *Sequencer64*, we created an new file-name to take its place, with a fall-back to the original file-name if the new file does not exist, or if *Sequencer64* is running in legacy mode.

After you run *Sequencer64* for the first time, it will generate a `sequencer64.usr` file in your home directory:

```
/home/ahlstrom/.config/sequencer64/sequencer64.usr
```

It allows you to give an alias to each MIDI bus, MIDI channel, and MIDI control codes, per channel. The name is a bit misleading... do not confuse this file with the `sequencer64.rc` file.

The process for setting up the user 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.

2. Define all of the instruments and their control-code 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.

So, taking our list of devices and channels we created above, 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 it supports, we can assemble a "user" configuration file that makes the setup visible in *Sequencer64*.

Peruse the next couple of sections to understand a bit about the format of this file. Look at the example file in the `contrib` directory as well, to see the whole thing put together. Once you're satisfied, go to section 9.6 "Sequencer64 "usr" File / Results" on page 125, and see what it all looks like.

9.1 Sequencer64 "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 `sequencer64.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 annoted example of one such section:

```
[user-midi-bus-0]
2x2 A (SuperNova,Q,TX81Z,DrumStation)     # name of the device
16                                        # number of channels

# NOTE: Channels are 0-15, not 1-16.  Instruments set to -1 = GM

0 1                                       # channel and instrument
1 1
2 1
3 1
4 1
5 1
6 1
7 1
8 3
9 3
10 3
11 0
12 0
13 0
14 0
15 2
```

Here's an example of one that needs only one override:

```
[user-midi-bus-2]
PCR-30 (303)
1                                       # number of channels
0 8                                     # channel and instrument
# The rest default to -1 - General MIDI
```

Please note that, as of version 0.9.10.1, these sections are read from the "user" configuration file only if the `--reveal-alsa-ports` option is *off*. Otherwise, the actual port names reported by ALSA are shown. The `user-midi-bus-definitions` and `user-midi-bus-N` sections are misleading if one wants to have access to the actual ALSA ports that exist on the system. Therefore, if the `--reveal-alsa-ports` option is turned on, then the definitions in the "user" configuration file are *not* read from that file.

9.2 Sequencer64 "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.

```
[user-instrument-definitions]
9     # number of user instrument
```

So this "usr" file will define 9 instruments. We will provide one section as a sample.

```
[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
5 Glide Rate
6
7 Channel Volume
8
9
10 Pan
11
12 Arp Range (0-9) (1-10 octaves)
13 Arp Length (0-15) (1-16 steps)
14 Arp Active (0-3) (Off,On,One Shot,Hold)
15 LFO 1 Shape (0-5) (Sine,Tri,Square,Saw,Rand,S&H)
   . . .
119
120 All Sound Off (0)
121 Reset All Controllers (0)
```

```
122 Local Control (0-127) (Off,On)
123 All Notes Off (0)
124
125
126
127
```

We assume that an unnamed control number is an unsupported control number.

Here is an instrument where its synthesis parameters can be controlled:

```
[user-instrument-1]
SuperNova
128
0 Bank Select MSB
1 Modulation Wheel
2 Breath Controller
3 Arp Pattern Select
4 Ring Modulator 2 * 3 Mix Level
5 Portamento Time
6 Data Entry
7 Part / Program Volume
8 Effects Confg Morph Amount
9 Arp Speed (Internal Clock Rate) [*]
10 Pan
11 Osc 1 Fine Tune
12 Osc 3 Fine Tune
13 Osc 1 Soften
14 Osc 2 Soften
15 Osc 3 Soften
16 LFO 1 Speed
17 LFO 1 Delay
   . . .
119 Delay Mod Wheel Depth
120 All Sound Off
121 Reset Controllers
122 Local Control [*]
123 All Notes Off
124 All Notes Off
125 All Notes Off
126 All Notes Off
127 All Notes Off
```

Here is an instrument that perhaps has no controllers, or maybe is simply not configured yet.

```
[user-instrument-4]
```

```
WaveStation
0
```

The sample file `contrib/scripts/dot-seq24usr` contains examples of some other kinds of instruments, such as drum machines.

9.3 Sequencer64 "usr" File / User Interface Settings

This section begins with an "INI" group marker `[user-interface-settings]`.

It provides for a feature we will hopefully be able to complete some day: the absolute specificiation of the appearance of the user interface. There is plenty of room to change the appearance of *Sequencer64* already! Please try them and see what you like.

```
#   ======== Sequencer64-Specific Variables Section ========

[user-interface-settings]

# These settings specify the soon-to-be-modifiable sizes of
# the Sequencer64 user-interface elements.

# Specifies the style of the main-window grid of patterns.
# 0 = normal style, matches the GTK theme, has brackets.
# 1 = white grid boxes that have brackets.
# 2 = black grid boxes.
2       # grid_style



# Specifies box style box around a main-window grid of patterns.
# 0  = Draw a whole box around the pattern slot.
# 1  = Draw brackets on the sides of the pattern slot.
# 2 and up = make the brackets thicker and thicker.
# -1 = same as 0, draw a box one-pixel thick.
# -2 and lower = draw a box, thicker and thicker.
2       # grid_brackets



# Specifies the number of rows in the main window.
# At present, only a value of 4 is supportable.
# In the future, we hope to support an alternate value of 8.
4       # mainwnd_rows
```

```
# Specifies the number of columns in the main window.
# At present, only a value of 8 is supportable.
8       # mainwnd_cols



# Specifies the maximum number of sets, which defaults to 1024.
# It is currently never necessary to change this value.
32      # max_sets



# Specifies the border width in the main window.
0      # mainwid_border



# Specifies the border spacing in the main window.
2      # mainwid_spacing



# Specifies some quantity, it is not known what it means.
0      # control_height



# Specifies the initial zoom for the piano rolls.  Ranges from 1.
# to 32, and defaults to 2 unless changed here.
2      # zoom



# Specifies if the key, scale, and background sequence are to be
# applied to all sequences, or to individual sequences.  The
# behavior of Seq24 was to apply them to all sequences.  But
# Sequencer64 takes it further by applying it immediately, and
# by saving to the end of the MIDI file.  Note that these three
# values are stored in the MIDI file, not this configuration file.
# Also note that reading MIDI files not created with this feature
# will pick up this feature if active, and the file gets saved.
# It is contagious.
#
# 0 = Allow each sequence to have its own key/scale/background.
#     Settings are saved with each sequence.
# 1 = Apply these settings globally (similar to seq24).
#     Settings are saved in the global final section of the file.
1      # global_seq_feature
```

```
# Specifies if the old, console-style font, or the new anti-
# aliased font, is to be used as the font throughout the GUI.
# In legacy mode, the old font is the default.
#
# 0 = Use the old-style font.
# 1 = Use the new-style font.
1      # use_new_font



# Specifies if the user-interface will support two song editor
# windows being shown at the same time.  This makes it easier to
# edit songs with a large number of sequences.
#
# 0 = Allow only one song editor (performance editor).
# 1 = Allow two song editors.
1      # allow_two_perfedits



# Specifies the number of 4-measure blocks for horizontal page
# scrolling in the song editor.  The old default, 1, is a bit
# small.  The new default is 4.  The legal range is 1 to 6, where
# 6 is the width of the whole performance piano roll view.
4      # perf_h_page_increment



# Specifies the number of 1-track blocks for vertical page
# scrolling in the song editor.  The old default, 1, is a bit
# small.  The new default is 8.  The legal range is 1 to 18, where
# 18 is about the height of the whole performance piano roll view.
8      # perf_v_page_increment



# Specifies if the progress bar is colored black, or a different
# color.  The following integer color values are supported:
#
# 0 = black
# 1 = dark red
# 2 = dark green
# 3 = dark orange
# 4 = dark blue
# 5 = dark magenta
# 6 = dark cyan
6      # progress_bar_colored
```

```
# Specifies if the progress bar is thicker.  The default is 1
# pixel.  The 'thick' value is 2 pixels.  (More than that is not
# useful.  Set this value to 1 to enable the feature, 0 to disable
# it.
1      # progress_bar_thick



# Specifies using an alternate (darker) color palette.  The
# default is the normal palette.  Not all items in the user
# interface are altered by this setting, and it's not perfect.
# Set this value to 1 to enable the feature, 0 to disable it.
0      # inverse_colors



# Specifies the window redraw rate for all windows that support
# that concept.  The default is 40 ms.  Some windows used 25 ms.
40     # window_redraw_rate



# Specifies using icons for some of the user-interface buttons
# instead of text buttons.  This is purely a preference setting.
# If 0, text is used in some buttons (the main window buttons).
# Otherwise, icons are used.  One will have to experiment :-).
0      # use_more_icons (currently affects only main window)
```

Note that the window-redraw rate option is meant more for experimentation than anything else. It probably doesn't affect CPU usage much, but might provide a smoother-running cursor on some systems.

9.4 Sequencer64 "usr" File / User MIDI Settings

This section begins with an "INI" group marker `[user-midi-settings]`.

It provides for a feature we will fully support (we are really close): Being able to support files with different PPQN, and to specify the global defaults for tempo, beats per measure, and so on.

```
[user-midi-settings]

# These settings specify MIDI-specific value that might be
# better off as variables, rather than constants.
# Specifies parts-per-quarter note to use, if the MIDI file.
# does not override it.  Default is 192, but we'd like to go
# higher than that.  BEWARE:  STILL GETTING IT TO WORK!
192     # midi_ppqn
```

```
# Specifies the default beats per measure, or beats per bar.
# The default value is 4.
4       # midi_beats_per_measure/bar



# Specifies the default beats per minute.  The default value
# is 120, and the legal range is 1 to 600.
120     # midi_beats_per_minute



# Specifies the default beat width. The default value is 4.
4       # midi_beat_width



# Specifies the buss-number override. The default value is -1,
# which means that there is no buss override.  If a value
# from 0 to 31 is given, then that buss value overrides all
# buss values specified in all sequences/patterns.
# Change this value from -1 only if you want to use a single
# output buss, either for testing or convenience.  And don't
# save the MIDI afterwards, unless you really want to change
# all of its buss values.
-1      # midi_buss_override
```

For the new 0.90 series, additional values for the `[user-midi-settings]` section have been added:

```
# Specifies the default velocity override when adding notes in the
# sequence/pattern editor.  This value is obtained via the 'Vol'
# button, and ranges from 0 (not recommended :-) to 127.  If the
# value is -1, then the incoming note velocity is preserved.
80      # velocity_override (-1 = 'Free')



# Specifies the precision of the beats-per-minutes spinner and
# MIDI control over the BPM value.  The default is 0, which means
# the BPM is an integer.  Other values are 1 and 2 decimal digits
# of precision.
1       # bpm_precision
```

```
# Specifies the step increment of the beats/minute spinner and
# MIDI control over the BPM value.  The default is 1. For a
# precision of 1 decimal point, 0.1 is a good value.  For a
# precision of 2 decimal points, 0.01 is a good value, but one
# might want somethings a little faster, like 0.05.
0.1    # bpm_step_increment


# Specifies the page increment of the beats/minute field. It is
# used when the Page-Up/Page-Down keys are pressed while the BPM
# field has the keyboard focus.  The default value is 10.
5.0    # bpm_page_increment
```

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.

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.

To obtain these new settings, remember to back up the *sequencer64.usr*, then run *Sequencer64* with the `--user-save` option, and then do a "diff" on the new file and the original to merge any old value that need to be preserved. Then make any further tweaks to the new values.

9.5 Sequencer64 "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 we are running out of single-character options. This group of options current provides the options shown below.

```
# The daemonize option is used in seq64cli to indicate that the
# application should be gracefully run as a service.
0       # option_daemonize
```

If this option is not used when running `seq64cli`, 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 `seq64cli` is to send a signal (e.g. SIGKILL) to it, or use MIDI control.

```
# This value specifies an optional log-file that replaces output
# to standard output and standard error.  To indicate no log-file,
# the string "" is used.
"seq64.log"
```

This log-file is written to the same directory as the "rc" and "usr" files.

9.6 Sequencer64 "usr" File / Results

Okay, now we have this file copied to our home directory:

```
/home/ahlstrom/.config/sequencer64/sequencer64.usr
```

If we'd already run *Sequencer64* at least once, we'd have overwritten the skeleton sample file that *Sequencer64* writes by default. We now have a full-fledged "user" file.

However, because we don't actually have all that equipment (we got the example from the Web, for cryin' out loud), let's see what we end up with when we run *Sequencer64* this time.

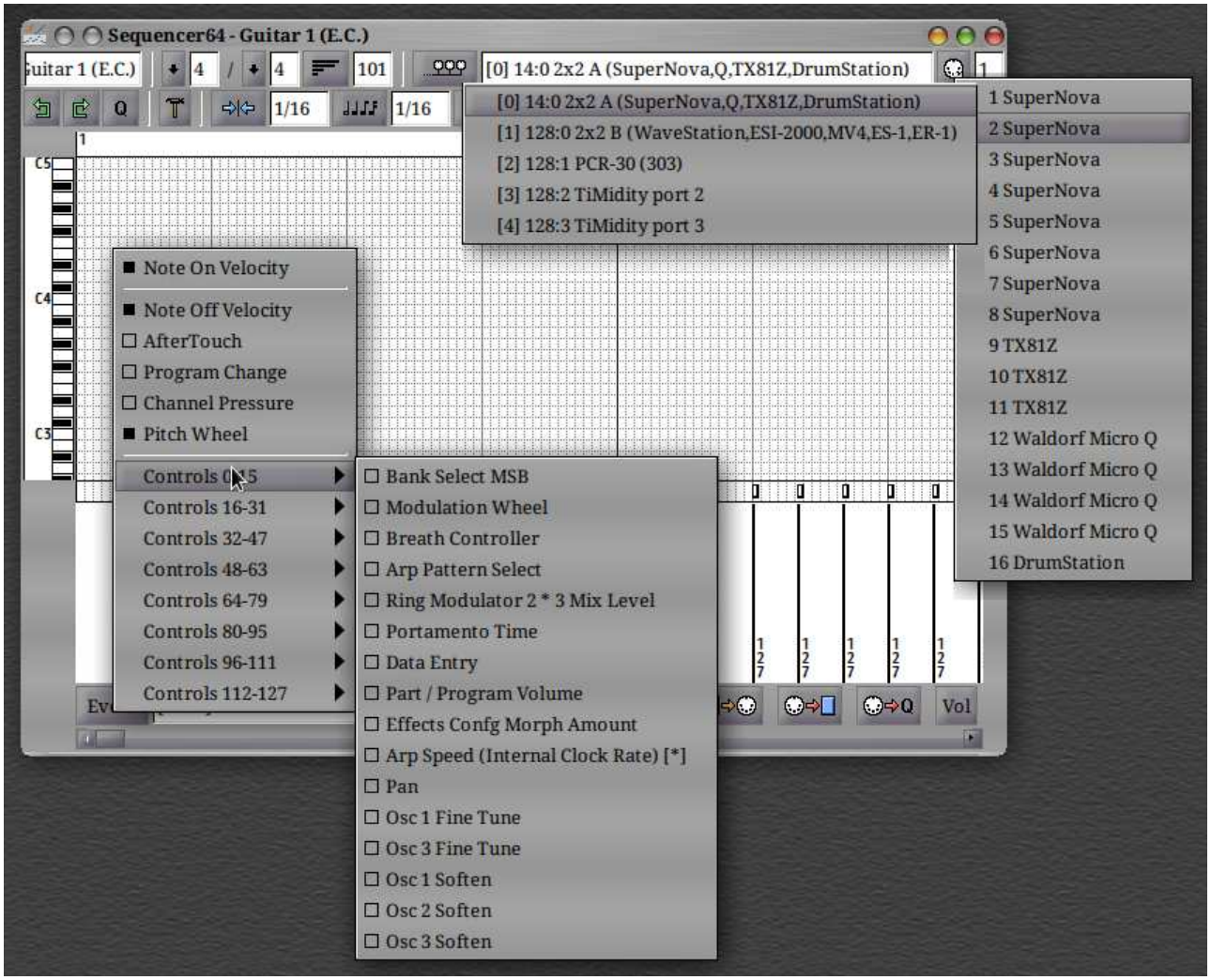


Figure 76: Sequencer64 Composite View of Non-Native Devices

Compare that diagram to figure 75 "Sequencer64 Composite View of Native Devices" on page 114. If the original figure, we saw the 5 native busses (ports) on our system, their bare-bones channel numbers, and the default controller values. In this new figure, we see the three buss devices (ports), plus the two Timidity ports. If we stopped the Timidity service, these would go away.

Look at the selected buss, "[0]". It's 16 channels are now associated with the devices to which the channels have been assigned.

Thus, when we have a new pattern we've created in *Sequencer64*, can assign it to exactly the buss and device we want.

If we don't have port-mappers installed, and thus have only one playback device plugged into the buss, we can still create a setup that shows the device and a specific program setup. Doing so would be tedious, but perhaps there's some automated way to do it?

Lastly, note the following figure.

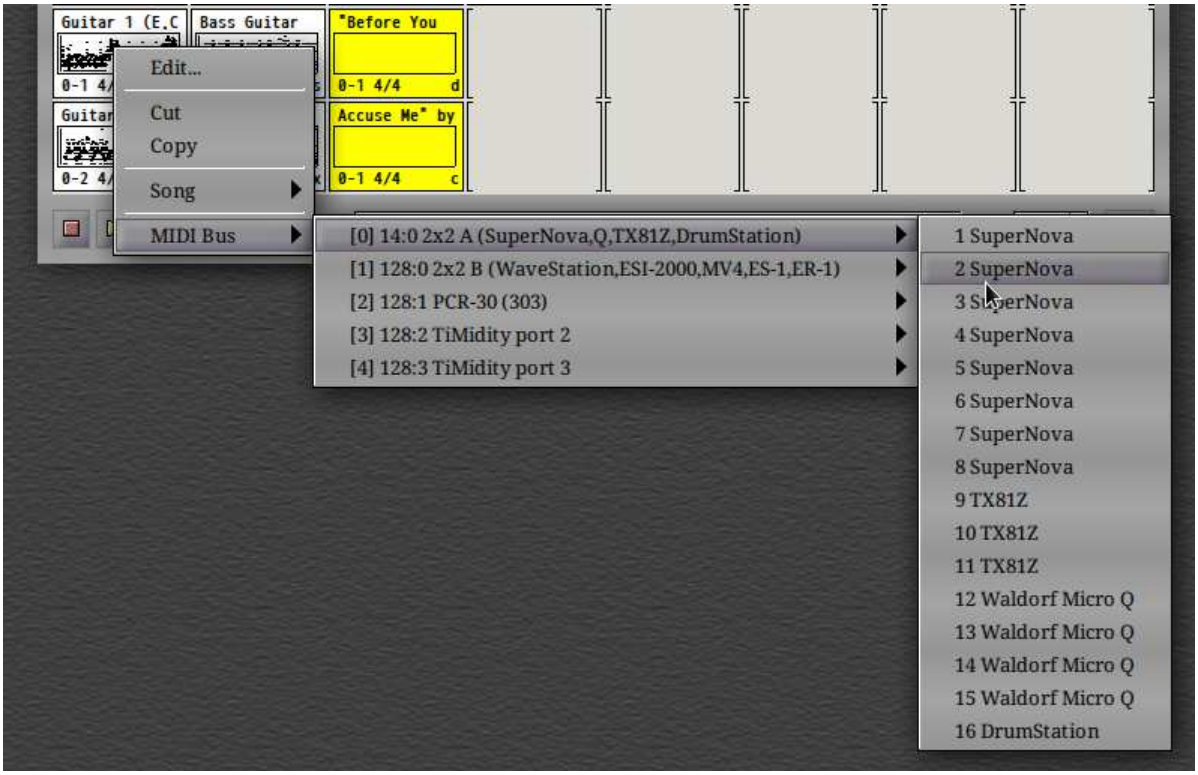


Figure 77: The MIDI Bus Menu for a Specific Pattern

This figure shows that we can also select the desired port and channel directly from the main window.

There's a lot more to the "user" configuration file than we've exposed here, but finding more information about this file has proven a bit tricky.

Sometime we would like to create a "user" that sets up the *Yoshimi* 1.3.5+ software synthesizer as a device and instrument.

10 Sequencer64 Man Page

This section presents the contents of the *Sequencer64* man page, but not exactly in *man* format. Also, an item or two are shown that somehow didn't make it into the man page, and minor corrections and formatting tweaks were made. For example, we replaced the underscore with the hyphen in the names

of some options. The legacy Seq24 options, which use underscores or are missing the option hyphen, are still unofficially supported.

`$HOME/.config/sequencer64/sequencer64.rc` holds the "rc" settings for *Sequencer64*.

`$HOME/.config/sequencer64/sequencer64.usr` holds the "user" settings for *Sequencer64*.

But the old style names are used for the "legacy" mode. See the `--legacy` option below. Here is the basic command line:

```
sequencer64 [OPTIONS] [FILENAME]
seq64 [OPTIONS] [FILENAME]
```

Sequencer64 accepts the following options, plus an optional name of a MIDI file. Please note that many of the options are 'sticky'. If they are used on the command-line, their settings are saved to the configuration files when *Sequencer64* exits.

`-h --help`
Display a list of all command-line options, then exit.

`-v --version`
Display the program version, then exit.

`-H --home [directory]`
New: Change the "home" directory from `.contrib/sequencer64` (always relative to `$HOME`. This option causes the `sequencer64.rc` and `sequencer64.usr` files to be loaded from or saved to a different directory. Format: `--home dirname`.

`-l --legacy`
New: Save the MIDI file in the old Seq24 format, as unspecified binary data, instead of as a legal MIDI track with meta events. Also read the configuration, if provided, from the "legacy" `/.seq24rc` and `/.seq24usr` files.

The user-interface will indicate this mode with a small text note. This mode is also used if *Sequencer64* is invoked as the `seq24` command (one can create a soft link to the sequencer64 executable to make that happen).

`-b --bus [buss]`
New: Supports modifying the buss number on all tracks when a MIDI file is read. All tracks are loaded with this buss-number override. This feature is useful for testing, making it easy to map the MIDI file onto the system's current hardware/software synthesizer setup. Also note that this option applies the MIDI buss override to any new sequences, as well. Format: `--bus bussnumber` or `--buss bussnumber`.

`-q --ppqn [ppqn]`
New: Supports modifying the PPQN value of Sequencer64, which is currently hardwired to a value of 192. This setting should allow MIDI files to play back at the proper speed, and be written with the new PPQN value. This feature is basically done. It seems to work – one can load MIDI files of arbitrary PPQN, and they play normally and look normal in the editor windows. They can also be saved, and load with the new PPQN value. But use the feature carefully for now, and save your work. We'll have more to say on this topic in the future. Format: `--ppqn ppqnnumber`.

`-L --lash`
New: If LASH support is compiled into the program, this option enables it. If LASH support is not compiled into the program, this option will not be shown in the output of the –help option.

`-n --no-lash`
New: If LASH support is compiled into the program, this option disables it, even if the default or configuration file set it. If LASH support is not compiled into the program, this option will not be shown in the output of the –help option.

`N/A --file [filename]`
Load a MIDI file on startup. **Bug:** This option does not exist. Instead, specify the file itself as the last command-line argument.

`-m --manual-alsa-ports`
Sequencer64 won't attach the system's existing ALSA ports. Instead, it will create is own set of input and output busses/ports.

`-a --auto-alsa-ports`
Sequencer64 will attach the system's existing ALSA ports. This variant is useful for overriding the rc configuration file.

`-r --reveal-alsa-ports`
New: *Sequencer64* will show the names of the ALSA port that the system defines, rather than the names defined in the 'user' configuration file.

`-R --hide-alsa-ports`
Sequencer64 will show the names of the ALSA port that the 'user' configuration file define, rather than the names defined by ALSA.

`-A --alsa`
Sequencer64 will not run the JACK support, even if specified in the configuration file. The configuration options are sticky (they are saved), and sometimes they aren't what you want to run.

`-s --show-midi`
Dumps incoming MIDI to the screen.

`-p --priority`
Runs at higher priority with a FIFO scheduler.

`N/A --pass-sysex`
Passes any incoming SYSEX messages to all outputs. Not yet supported.

`-i --ignore [number]`
Ignore ALSA device [number].

`-k --show-keys`
Prints pressed key value.

`-K --inverse`
Changes the color scheme for the sequence editor and performance editor piano rolls. It basicially inverts the colors. It can be considered kind of a "night mode".

`-x --interaction-method [number]`
Select the mouse interaction method. 0 = seq24 (the default); and 1 = fruity loops method. The latter does not completely support all actions supported by the Seq24 interaction method, at this time.

The following options will not be shown by –help if the application is not compiled for JACK support.

`-j --jack-transport`
Sequencer64 will sync to JACK transport.

`-J --jack-master`
Sequencer64 will try to be JACK master.

`-C --jack-master-cond`
JACK master will fail if there is already a master.

`-M --jack-start-mode [x]`
When *Sequencer64* is synced to JACK, the following play modes are available: 0 = live mode; and 1 = song mode, the default.

`-S --stats`
Print statistics on the command-line while running. Not available unless this option has been compiled in at build time, which can be determined by using the `--version` option.

`-U --jack-session-uuid [uuid]`
Set the UUID for the JACK session.

`-u --user-save`
Save the "user" configuration file when exiting Sequencer64. 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. Note that the "rc" configuration option are generally also saved. But see the "auto-option-save" directive in the "rc" file. It is new with version 0.9.9.15.

`-f --rc filename`
Use a different "rc" configuration file. It must be a file in the user's $HOME/.config/sequencer64 directory or the directory specified by the –home option. Not supported by the –legacy mode. The '.rc' extension is added if no extension is present in the filename.

`-F --usr filename`
Use a different "usr" configuration file. It must be a file in the user's $HOME/.config/sequencer64 directory or the directory specified by the –home option. Not supported by the –legacy mode. The '.usr' extension is added if no extension is present in the filename.

`-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`. Makes the `seq64cli` application fork to the background.
- `no-daemonize`. Makes the `seq64cli` application run in the foreground, where it is easy to see the informational output written to the console.
- `log=filename`. Reroutes standard error and standard output messages to the given log-file. This file is located in the directory for the "rc" and "usr" files (which can be altered via the `-H/--home` directory option.

`$HOME/.config/sequencer64.rc` holds the main configuration settings for Sequencer64. If it does not exist, it will be generated when Sequencer64 exits. If it does exist, it will be rewritten with the current configuration of Sequencer64. Many, or most, of the command-line options are "sticky", in that they will be written to the configuration file.

`$HOME/.config/sequencer64.usr` stores the MIDI-configuration settings and some of the user-interface settings for Sequencer64. If it does not exist, it will be generated with a minimal configuration when

Sequencer64 exits. If it does exist, it will be rewritten with the current configuration of Sequencer64. Note that the `--legacy` option causes the old configuration-file names to be used.

The current Sequencer64 project homepage is a simple git repository at

https://github.com/ahlstromcj/sequencer64.git.

Up-to-date instructions can be found in the project at

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

The old Seq24 project homepage is at http://www.filter24.org/seq24/ the new one is at https://edge.launchpad.net/seq24/. It is released under the GNU GPL license. Sequencer64 is also released under the GNU GPL license.

Sequencer64 was written by Chris Ahlstrom ¡ahlstromcj@gmail.com¿ (with a fair amount of help). *Seq24* was written by Rob C. Buse mailto:seq24@filter24.org and the *Sequencer64* team.

This manual page was written by Dana Olson mailto:seq24@ubuntustudio.com with additions from Guido Scholz mailto:guido.scholz@bayernline.de and Chris Ahlstrom mailto:ahlstromcj@gmail.com.

Version 0.9.9.6 November 4 2015 sequencer64(1)

11 Concepts

The *Sequencer64* program is basically a loop-playing machine with a fairly simple interface. Before we describe this interface, it is useful to present some concepts and definitions of terms as they are used in *Sequencer64*. Various terms have been used over the years to mean the same thing (e.g. "sequence", "pattern", "loop", and "slot"), so it is good to clarify the terminology.

11.1 Concepts / Terms

This section doesn't provide comprehensive coverage of terms. It covers terms that puzzled the author at first or that are necessary to understand the *Sequencer64* program.

11.1.1 Concepts / Terms / armed

An armed sequence is a sequence (see section 11.1.17 "Concepts / Terms / sequence" on page 133) that will be heard. "Armed" is the opposite of "muted". Performing an *arm* operation in *Sequencer64* means clicking on an "unarmed" sequence in the patterns panel (the main window of *Sequencer64*). An unarmed sequence will not be heard, and it has a white background. When the sequence is *armed*, it will be heard, and it has a black background. A sequence can be armed or unarmed in three ways:

- Clicking or Shift-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.

11.1.2 Concepts / Terms / buss (bus)

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. A *buss* is just another name for port. See section 11.1.12 "Concepts / Terms / port" on page 132.

11.1.3 Concepts / Terms / export

A *export* in *Sequencer64* is a way of writing a song-performance to a more standard MIDI file, so that it can be played 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.

11.1.4 Concepts / Terms / group

A *group* in *Sequencer64* is one of up to 32 previously-defined mute/unmute patterns in the active screen set. A group is a set of patterns, in the current screen-set, that can arm (unmute) their playing state together. Every group contains all 32 sequences in the active screen set. This concept is similar to mute/unmute groups in hardware sequencers. Also known as a "mute-group".

11.1.5 Concepts / Terms / loop

Loop is a synonym for *pattern* or *sequence*, when used in existing *Seq24* documentation. Each loop is represented by a box (pattern slot) in the Pattern (main) Window.

11.1.6 Concepts / Terms / measures ruler

The *measures ruler* is the bar at the top of the Pattern Editor and Song Editor windows that shows the numbering of the measures in the song. Left, right, or end markers can be dropped on this ruler to set durations to be played, looped, expanded, or collapsed.

Note: The original *Seq24* documentation calls this item the *bar indicator*.

11.1.7 Concepts / Terms / event strip

The *event strip* is the bar at the bottom of the Pattern Editor window that shows the location of events in the pattern. For Note On and Note Off events, it is shown in gray to warn the user to be careful in moving these events.

11.1.8 Concepts / Terms / muted

The opposite of section 11.1.1 "Concepts / Terms / armed" on page 130.

11.1.9 Concepts / Terms / MIDI clock

MIDI clock is a MIDI timing reference signal used to synchronize pieces of equipment together. MIDI clock runs at a rate of 24 ppqn (pulses per quarter note). This means that the actual speed of the MIDI clock varies with the tempo of the clock generator (as contrasted with time code, which runs at a constant rate).

11.1.10 Concepts / Terms / pattern

A *Sequencer64 pattern* (also called a "sequence" or "loop") is a short unit of melody or rhythm in *Sequencer64*, extending for a small number of measures (in most cases). Each pattern is represented by a box in the Patterns window.

Each pattern is editable on its own. All patterns can be layed out in a particular arrangement to generate a more complex song.

pattern is a synonym for *loop* or *sequence*. It is our preferred term.

11.1.11 Concepts / Terms / performance

In the jargon of *Sequencer64*, a *performance* is an organized collection of patterns. This layout of patterns is created using the Song Editor, sometimes called the "performance editor". This window controls the song playback in "Song Mode". The playback of each track is controlled by a set of triggers created for that track.

11.1.12 Concepts / Terms / port

A *port* is just another name for buss. See section 11.1.2 "Concepts / Terms / buss (bus)" on page 131. Each port can support 16 MIDI channels.

11.1.13 Concepts / Terms / pulses per quarter note

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", "MIDI clocks", and more.

While a full description of all these terms, and how they are calculated, is beyond the scope of this document, we will try to clarify the discussion when such confusion could be an issue.

11.1.14 Concepts / Terms / queue mode

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 toggles occur 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.

11.1.15 Concepts / Terms / replace

Replacement is a form of muting/unmuting. When the "replace" key is pressed while click a sequence, that sequence is unmuted, and all of the other sequences are muted.

11.1.16 Concepts / Terms / screen set

The *screen set* is a set of patterns that fit within the 8x4 grid of loops/patterns in the Patterns panel. *Sequencer64* supports multiple screens sets, up to 32 of them, and a name can be given to each for clarity. Some day *Sequencer64* will support an 8x8 grid and 64 patterns per screen set.

11.1.17 Concepts / Terms / sequence

Sequence is another synonym for *pattern*, used in some of the *Seq24* documentation. *Loop* is another synonym. Each sequence is represented by a box (pattern slot) in the Patterns window.

Note that many, for 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.

11.1.18 Concepts / Terms / snapshot

A *Sequencer64 snapshot* is simply 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, instead.)

11.1.19 Concepts / Terms / song

A *song* is a collection of patterns in a specific layout, as assembled via the Song Editor window. Also see 11.1.11

11.1.20 Concepts / Terms / trigger

A *trigger* is a small data structure that indicates when a sequence 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 in ways that the musician can lay out.

11.2 Concepts / Sound Subsystems

11.2.1 Concepts / Sound Subsystems / ALSA

ALSA is a sound and MIDI system for Linux, with components built into the Linux kernel. It is the main subsystem used by *Sequencer64*. The name of the library used to build *ALSA* projects is `libasound`. See reference [1].

11.2.2 Concepts / Sound Subsystems / PortMIDI

PortMIDI is a cross-platform API (applications programming interface) for MIDI. It seems to be used in the "portmidi" C++ modules included with the base source-code repository of *Seq24* available (for example) from Debian Linux. See reference [12] for the PortMIDI home page. Unfortunately, the code in the Debian package is not quite ready to build on Windows. We might fix that someday, though Windows is not a high priority.

The SubatomicGlue Windows port of *Seq24* (see reference [27]) bundles a version of the PortMIDI project with the source code for the port. It also provides a complete bundle of the other products (e.g. gtkmm 2.4) needed to build and run the project. (By the way, the Windows port is built with MingW, which provides the GNU compilers and tools. This is a good thing, as Visual Studio Community, though "free", is not "Free".)

11.2.3 Concepts / Sound Subsystems / JACK

JACK is a cross-platform (with an emphasis on Linux) API and infrastructure for making it easier to connect and reroute MIDI and audio event between various applications and hardware ports. See reference [6].

12 Building Sequencer64 From Source Code

The current packaging for Sequencer64 is primarily aimed at developers. But note that we're starting to add Debian packages to a new "Sequencer64 Packages" project ([26]). If one has packages built for other Linux distributions, let us know, and we can stick them there for others to use.

This section presents a how-to on building the various versions of *Sequencer64* from source code. It's actually pretty easy, with these instructions.

12.1 INSTALL

There are many build options. Some are modifiable via the normal GNU `configure` script method. Many more are modifiable by editing the source code to **define** and **undefine** certain macros. If you don't care about options, start here. If you want to see what options are available, skip to section 12.2.1 "Using More "configure" Options" on page 136, which has many details one can adjust.

There is currently no `configure` script... it must be created by using the `bootstrap` script, as the following instructions make clear.

Steps:

1. Preload any dependencies, as listed in section 12.3 "Sequencer64 Build Dependencies" on page 141. However, if some are missing, the configure script will tell you, or, at worst, a build error will.
2. Check-out the branch you want; normally you will be happy to get "master". Make a branch if you want to make changes.
3. From the top project directory, run the commands:

```
$ ./bootstrap
$ ./configure
```

4. For debugging without libtool getting in the way, just run one of the following commands, which will run the `configure` script, adding the `--enable-debug` and `--disable-shared` options to it.

```
$ ./bootstrap --enable-debug
$ ./bootstrap -ed
```

5. Run the make command:

```
$ make
```

6. To install *Sequencer64*, become root and run:

```
# make install
```

Note that one has to build the documentation and debian packaging separately, they are not part of the default build.

Please note that there are other things one can do to speed up the build process. Already noted above is the `--enable-debug` option. The following command will bootstrap the code and then configure for release mode, and greatly reduce the amount of compiler output:

```
$ ./bootstrap --enable-release
$ ./bootstrap -er
```

This option will run the following command:

```
$ ./configure --enable-silent-rules
```

It results in abbreviated out, which makes it easier to see any warnings that might pop up This anti-verbosity option can be overridden at "make" time:

```
$ make V=1
```

(Using `V=0` is another way to quiet down the build.)

Also note that the build can be sped up by telling make to use more cores. For example, if one has an 8-core system:

```
$ make -j 9
```

Of course, one can use fewer than the number of cores, if desired.

12.2 Options for Sequencer64 Features

Sequencer64 comes with options for the `configure` command and options represented by definable macros in the source code.

12.2.1 Using More "configure" Options

The following `configure` options can be specified on the command line:

1. `--enable-rtmidi`. This option can be bootstrapped directly using (for example) "`./bootstrap -er -rm`. However, this option is currently the default build. It creates the default version of *Sequencer64*, an executable named `seq64`. This version adds the ability to do JACK input/output using the native API... no more need to use `a2jmidid` to bridge from ALSA to JACK. It provides a JACK client that is separate from the pre-existing JACK-transport client. In addition, this executable will fall back to using ALSA if JACK is not running. Like the ALSA support, the JACK support has an auto-connect feature that can be disabled using the "manual" ("virtual-port") mode of seq64.

One other note. We term this version "rtmidi" because we originally used the RtMidi ([13]) project as the basis for the native JACK support, until we realized it does not really fit the usage model of *Sequencer64*. So we heavily refactored how it works, keeping a couple of key features, and keeping the moniker "rtimidi".

2. `--enable-cli`. This option can be bootstrapped directly using (for example) "`./bootstrap -er -cli`". It builds a command-line version of `seq64` that has the command name `seq64cli`. Currently, this application can only be controlled via MIDI controls set up in the [midi-control] section of the "rc" file. See section 8.1 "Sequencer64 "rc" File / MIDI Control Section" on page 96, for more information on those controls, which now also include start, stop, and pause commands. Also, currently the only way to load a MIDI file is as the last command-line argument. There is also an option to make the application fork into the background as a daemon. We're looking in to using OSC as a control mechanism, but, until then, this executable is of limited usefulness.

3. `--enable-alsamidi`. This option can be bootstrapped directly using (for example) "`./bootstrap -er -am`". This executable is basically the original version of *Sequencer64*, with the original executable name of `sequencer64`, which we're keeping around as a backup while we work the remaining nits out of the "rtmidi" version of the application.

4. `--enable-portmidi`. This option can be bootstrapped directly using (for example) "`./bootstrap -er -pm`". This option builds the PortMIDI version of the application, named `seq64portmidi`. This version works with Linux, but was meant as a way to port *seq24* to Windows ("Google" for "seq24 subatomic glue"). However, the *Sequencer64* project deprecates this version. Eventually, we hope to migrate in the Windows and Mac OSX support modules of the RtMidi ([13]) project.

5. `--disable-highlight`. This option undefines the `SEQ64_HIGHLIGHT_EMPTY_SEQS` macro, which is otherwise defined by default. If defined, the application will highlight empty sequences/patterns by coloring them yellow. If not defined, empty sequences/patterns are shown in the normal black-on-white coloring. In either case, empty patterns will not be played.

6. `--disable-lash`. This option undefines the `SEQ64_LASH_SUPPORT` macro, which is otherwise defined by default. Even if this option is left defined, however, *Sequencer64* will still not use LASH support unless one specifies `--lash` on the `sequencer64` command-line or turn on the new `[lash-session]` option in the "rc" configuration file, `/.config/sequencer64/sequencer64.rc`.

7. `--disable-jack`. This option undefines the `SEQ64_JACK_SUPPORT` macro, which is otherwise defined by default. Even if this option is left defined, however, *Sequencer64* will still not use JACK support unless one specifies the various JACK options on the `sequencer64` command-line or turn them on in the "rc" configuration file, `/.config/sequencer64/sequencer64.rc`.

8. `--disable-jack-session`. This option undefines the `SEQ64_JACK_SESSION` macro, which is defined if JACK support is defined, and the jack/session.h file is found. Again, this option, if left defined, can be affected by command-line options and options in the "rc" configuration file.

9. `--disable-pause`. This option undefines the `SEQ64_PAUSE_SUPPORT` macro, which is defined by default, and provides support for toggling between a Play button and a Pause button, for actually pausing playback in Live mode, and for a new Pause key, which defaults to a period (".").

10. `--disable-chords`. This option undefines the `SEQ64_STAZED_CHORD_GENERATOR` macro. If this macro is defined, then the application is built with support for a Chord button in the pattern editor, which enables entering whole chords with a single click. This feature is grabbed from *Seq32* ([21]).

11. `--disable-transpose`. This option undefines the `SEQ64_STAZED_TRANSPOSE` macro. If this macro is defined, then the application is built with support for a Transpose button in the pattern editor and the song editor. The Transpose button in the pattern editor allows a pattern to be exempt from transposition, while the Transpose button in the song editor allows transposing the entire song (except for exempt patterns). This feature is grabbed from *Seq32* ([21]).

To summarize, these options undefine the following build macros:

- `SEQ64_HIGHLIGHT_EMPTY_SEQS`
- `SEQ64_LASH_SUPPORT`
- `SEQ64_JACK_SUPPORT`
- `SEQ64_JACK_SESSION`
- `SEQ64_PAUSE_SUPPORT`
- `SEQ64_STAZED_CHORD_GENERATOR`

12.2.2 Manually-defined Macros in the Code

As we have explored what *Seq24* does while we add improvements to *Sequencer64*, we've found a lot of things that might change the code for the worse in some people's minds. So we've been careful to mark those changes with macros. And sometimes we tried a change, but left it disabled. You are free to look at those macros, modify them, and build the source code to one's preferences. If one does not see a macro described below, it means we need to catch up with the documentation.

The following items are not yet part of the configure script, but can be edited manually to achieve the desired settings:

1. `SEQ64_EDIT_SEQUENCE_HIGHLIGHT`. Already defined in the `perform` module. Provides the option to highlight the currently-editing sequence in the main window view and in the song editor. If the sequence is muted, it is highlighted in black text on a cyan background. If it is unmuted, it is highlighted in cyan text on a black background. The highlighting follows whichever pattern editor or event editor has the focus.

2. `SEQ64_USE_NEW_FONT`. Already defined in the `font` module. If defined, a new, anti-aliased, bold font is used in the user-interface. This new font is implemented in new XPM files in `resources/pixmaps` directory: `wen*.xpm`. The font is slightly larger, but changes the user-interface sizes only to an infinitesmal degree. Using this new font is the default.

Obsolete: This option is no longer a compile-time option, but a run-time option. It is now the default, but the usage of the old versus new font can be set in the "user" configuration file. Also, if the legacy mode is specified, the old font becomes the default.

3. `SEQ64_USE_EVENT_MAP`. Already defined in the `event_list` module. It enables the usage of an `std::multimap`, instead of an `std::list`, to store MIDI events. Because the code does a lot of sorting of events, using the `std::multimap` is actually a lot faster (especially under debug mode, where it takes many seconds for a medium-size MIDI file to load using the `std::list` implementation.

There is still a chance that the `std::multimap` might prove the limiting factor during playback. If that is the case, then we would probably implement dumping the multimap to a vector before playback. We shall see!

4. `SEQ64_USE_MIDI_VECTOR`**.** Already defined in the `midifile` module. It enables the usage of an `std::vector`, instead of an `std::list`, to store MIDI data bytes. It provides the preferred alternative to the list for storing and counting the bytes of MIDI data. It is an attempt to stop reversing certain events due to the peculiarities of using `std::list` to store MIDI bytes from a sequence. This new implementation uses `std::vector` and does not use `pop_back()` to retrieve the bytes for writing to a file.

5. `SEQ64_FOLLOW_PROGRESS_BAR`**.** Already defined in the `app_limits.h` module. It enables the automatic scrolling (horizontal paging) of the pattern editor and song editor piano rolls, to keep the progress bar in view at all times. This feature is useful for patterns that are longer than the span of the editor windows. Such scrolling is a common feature of software MIDI sequencers.

Obsolete: Still need to replicate the descriptions that follow in the proper sections.

6. `SEQ64_USE_GREY_GRID`**. This item is no longer defined**. Instead, the option is now part of the "rc" configuration file. This description will be moved to the correct section eventually.

This configuration item causes the pattern slots/boxes to be colored grey (actually, they will be colored normally as per the current GTK them). Otherwise, they are colored black. By default, this value is defined (in the `mainwid` module).

7. `SEQ64_USE_WHITE_GRID`**. This item is no longer defined**. Instead, the option is now part of the "user" configuration file. This description will be moved to the correct section eventually.

This configuration item causes the pattern slots/boxes to be colored white. Also definable(in the `mainwid` module).

8. `SEQ64_USE_BRACKET_GRID`**. This item is no longer defined**. Instead, the option is now part of the "user" configuration file. This description will be moved to the correct section eventually. This configuration box that outlines the pattern slots/boxes is painted over to convert the box to look like a pair of brackets. By default, this value is defined (in the `mainwid` module).

9. `SEQ64_SEQNUMBER_ON_GRID`**. This item is no longer defined**. Instead, the option is now part of the "rc" configuration file. This description will be moved to the correct section eventually.

If the "show sequence numbers" option is on, then each of the blank pattern slots in the main window show the would-be sequence number for that slot. The background color of the numbers will not match the background color of the grid (which matches the chosen GTK theme). But, no matter what the GTK background color, they will at least be visible. There is a little image of this style inside the screenshot shown on the first page of this manual.

If `SEQ64_USE_WHITE_GRID` are defined, so that the grid cells are white, then the sequence numbering looks a little nicer, as can be seen in the following figure:

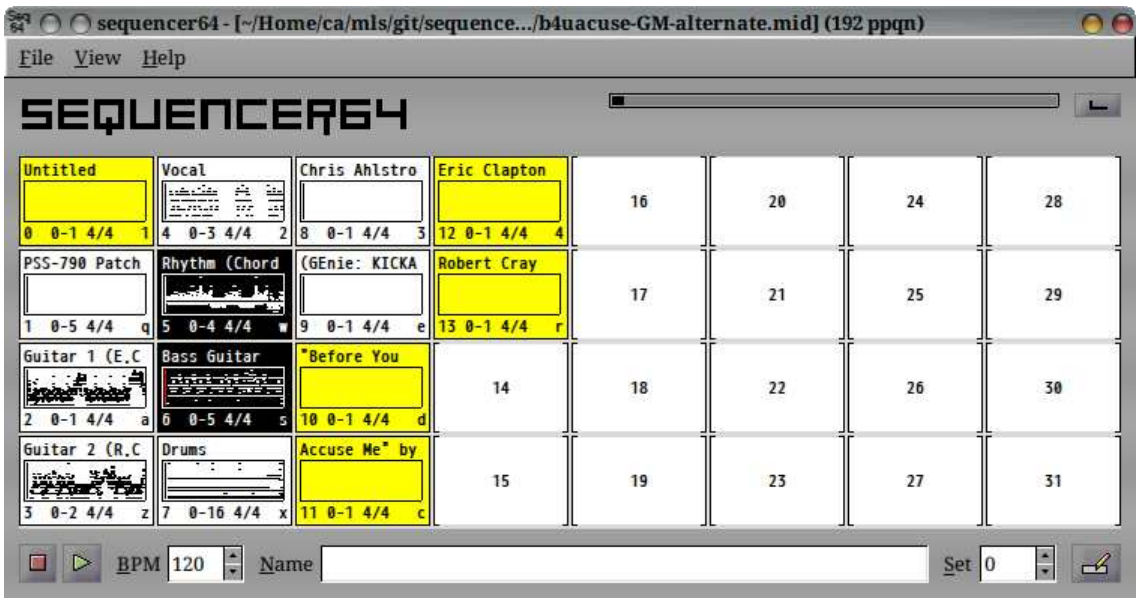


Figure 78: Pattern Window Built for White Grid with Numbering

There is a little image of this style inside the screenshot shown on the first page of this manual, as well.

If neither `SEQ64_USE_GREY_GRID` nor `SEQ64_USE_WHITE_GRID` are defined, so that the grid slots are black, then the numbering will be yellow on a black background, and match perfectly. This style is shown in the following figure:

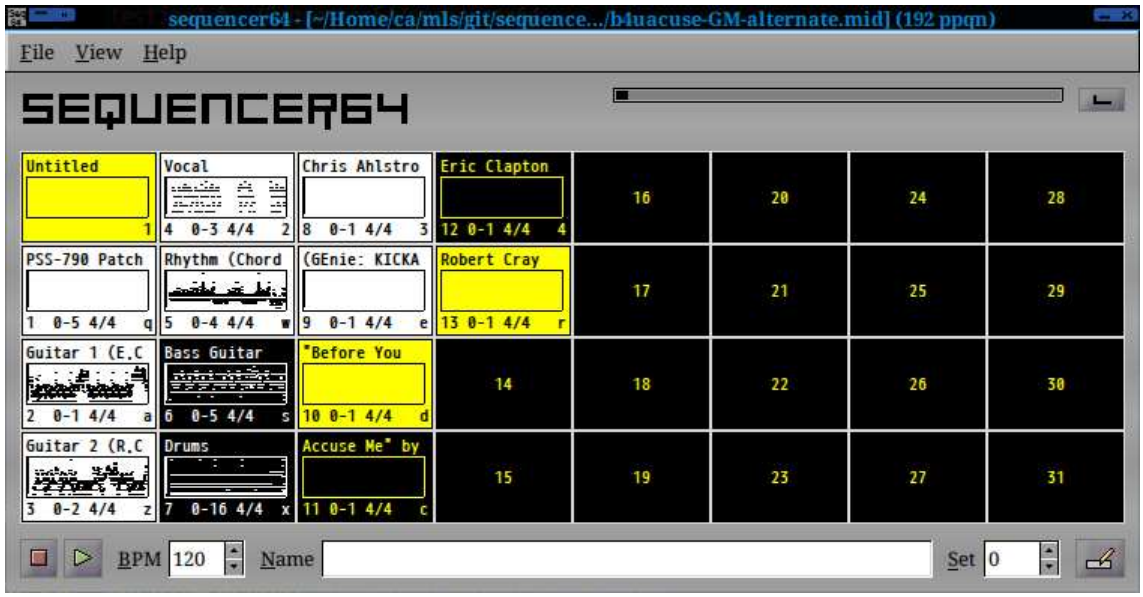


Figure 79: Pattern Window Built for Black Grid with Numbering

There is a little image of this style inside the screenshot shown on the first page of this manual, as well.

Take your pick, modify the code accordingly before building it. Perhaps these can eventually be options for the `configure` script, or even run-time options! Let us know!

10. `SEQ64_SOLID_PIANOROLL_GRID`**.** Enabling this macro makes the grid lines for the piano rolls more solid, with about the same perception of lightness. It also calls in some other tweaks, such as the positioning of markers. We currently like this look a little better, and so it is the default. See the `app_limits.h` header file for the definition of this variable.

Here is the pattern editor (sequence editor) with this alternate look.

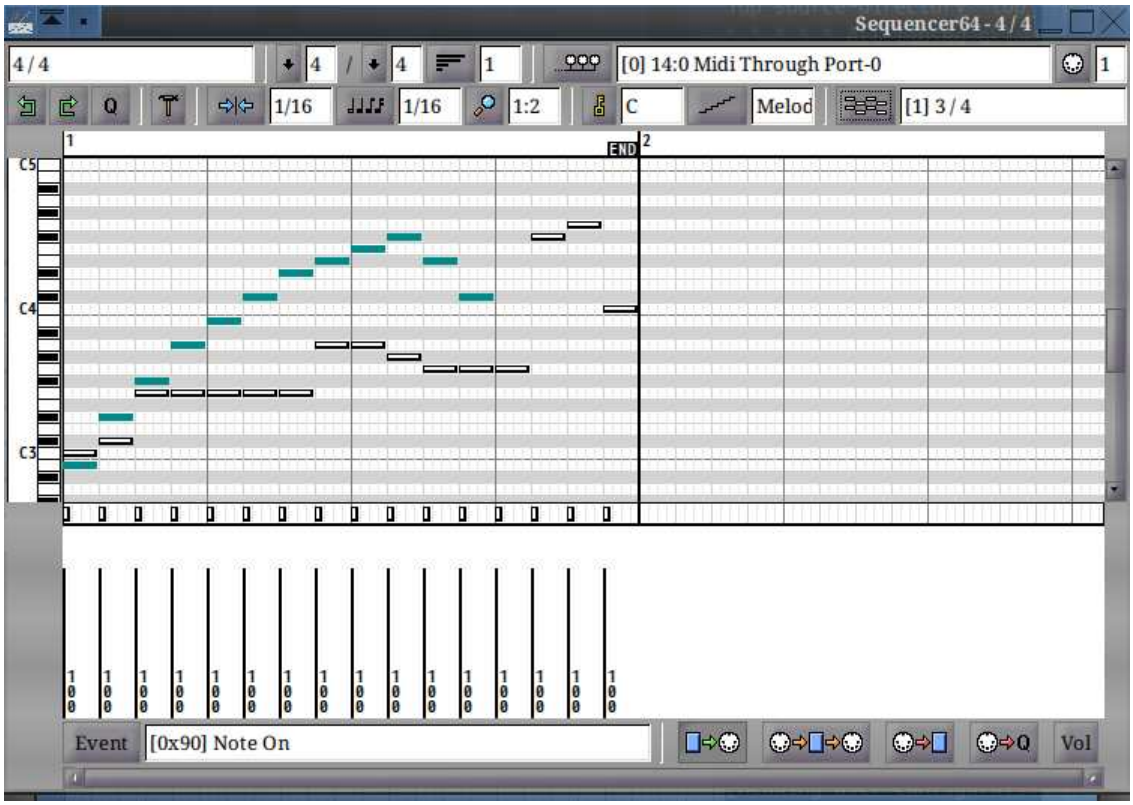


Figure 80: Sequence Pattern Editor Alternate Look

Note the smmoothness of the grid lines, the extra emphasis of the C notes at each octave, the emphasis of the note-drawing snap lines that mark the default length of a click-to-add note, the emphasis of the beat and bars, and, finally, the new location of the **END** marker. Also note the dark-cyan background pattern, discussed elsewhere in this document.

Here is the grid-styling for an 8/4 time signature in the song editor:

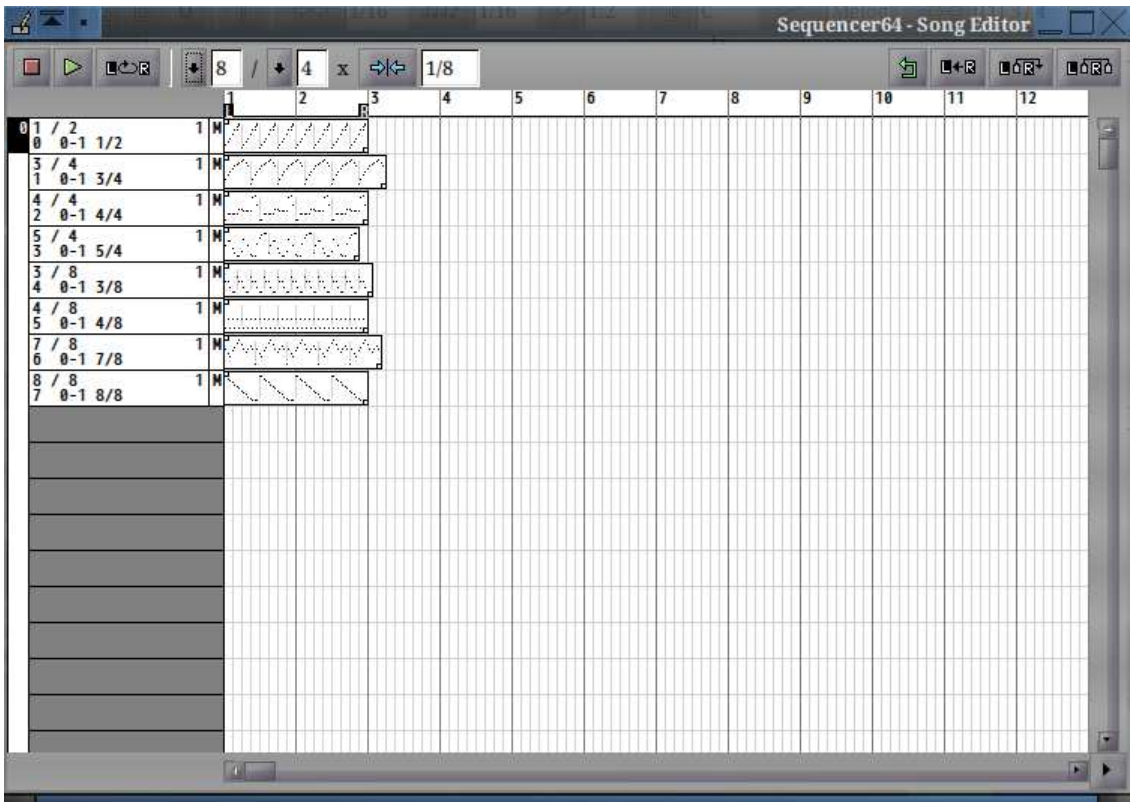


Figure 81: Song Editor Alternate Look

Also note the sequence numbers shown in the bottom left of each pattern name box. This is a new feature, and, as noted elsewhere, is a new option in the *File / Options / Keyboard* tab and in the "rc" configuration file.

11. `SEQ64_USE_VI_SEQROLL_MODE`**.** Definable in the seqroll module, this macro allows the vi hjkl keys to be used as arrow keys for moving notes. Not yet tested. We will not make this a default, because it could drive non-vi users nuts.

12. `SEQ64_USE_DEBUG_OUTPUT`**.** Enable this macro in the globals.h header file, to see extra console output if the application is compiled for debugging. This macro can be activated only if `PLATFORM_DEBUG` is defined, which is taken care of by the build process. If set, this macro turns on extra console output for the following modules:

- `globals`
- `jack_assistant`
- `optionsfile`
- `user_settings`

12.3 Sequencer64 Build Dependencies

With luck, the following dependencies will bring in their own dependencies when installed.

Code:

- libgtkmm-2.4-dev (dev is the header-file package)

- libsigc++-2.0-dev
- libjack-jackd2-dev
- liblash-compat-dev (optional)

Runtime:

- libatk-adaptor (and its dependencies)
- libgail-common (and its dependencies)
- valgrind (optional, very useful for debugging)
- gdb (optional, very useful for debugging)
- gprof and gcov (optional, very useful for debugging)

Build tools:

- automake and autoconf
- autoconf-archive
- g++
- make
- libtool
- More?

Documentation:

- doxygen and doxygen-latex
- graphviz
- texlive
- More?

Debian packaging:

- debhelper
- fakeroot
- More?

13 MIDI Format and Other MIDI Notes

13.1 Standard MIDI Format 0

New: *Sequencer64* can now read and import SMF 0 MIDI files, and performs channel splitting automatically.

When an SMF 0 format is detected, *Sequencer64* reads the file as if were an SMF 1 file, but puts all of the events into the same sequence/pattern. While the file is being processed, a list of the channels present in the track is maintained.

Tempo and Time Signature events are also read, if present in the file. (This new feature also holds for SMF 1 songs. In addition, when saving a *Sequencer64* MIDI file, in non-legacy mode, the Tempo and Time

Signature events are now also saved as MIDI events. This allows other sequencers to more thoroughly read a *Sequencer64* MIDI file.)

This addition of Tempo can also fix imported tracks that don't have a measure value (e.g. it has 0 instead of at least one measure) associated with them; unfixed, these tracks have racing progress bars that don't reflect the actually progresss through the track.

Once the end-of-track is encountered for that sequence, one new empty sequence is created for each channel found in the original (main) sequence. The events in the main sequence are scanned, one by one, and added at the end of the appropriate sequence. If the event is a channel event, then the event is inserted into the sequence that was created for that channel. If the event is a non-channel event, then each sequence 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 following figure shows in imported SMF 0 tune, split into tracks.

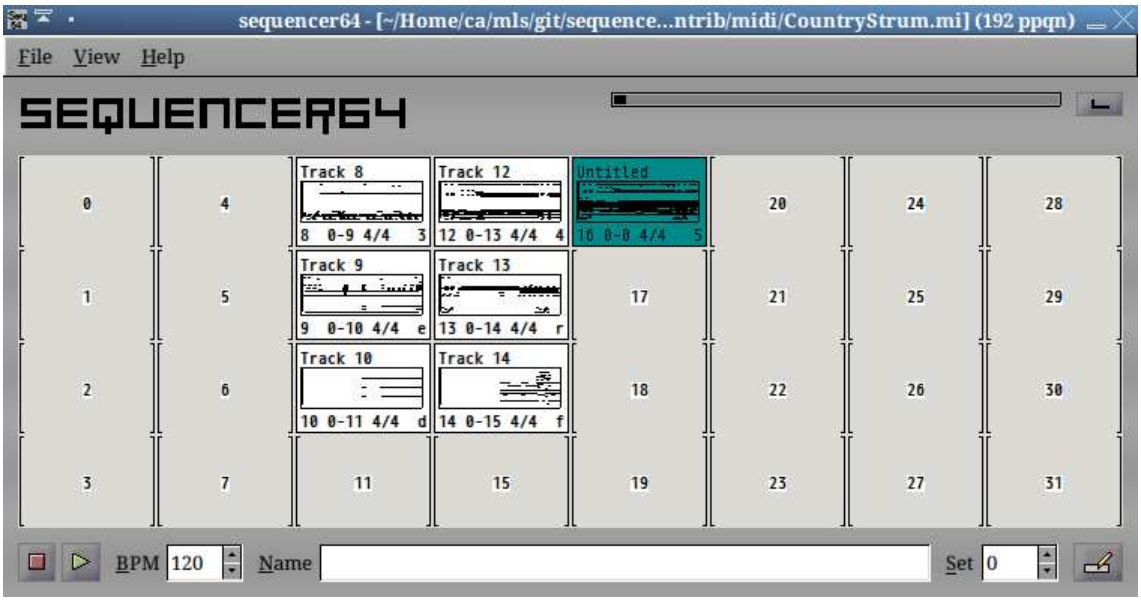


Figure 82: Imported SMF 0 MIDI Song

The original imported SMF 0 track is preserved, intact, in main window pattern slot #16. It is highlighted in a dark cyan color to remind the user that it is not a normal, playable *Sequencer64* sequence. It has no channel number. It is assigned the non-existent MIDI channel of 0. If the original track had no title, this track is named "Untitled". Normally, one will either delete this track before saving the file, or at least keep it muted.

Each new, single-channel track is given a title of either the form "N: Track-name" or, if the song was untitled, "Track N". The sequence number of each new track is the internal channel number (which is always the actual MIDI channel number minus one).

The time-signature of each track is currently set to defaults, unless a time-signature event is encountered in the imported file.

New: *Sequencer64* adds support for obtaining some of the other information a MIDI SMF 0 track might have, such as the Tempo and the Time Signature. It also now will save this information in the first track of the MIDI file.

The original SMF 0 track is also shown in the song editor, as shown in the following figure.

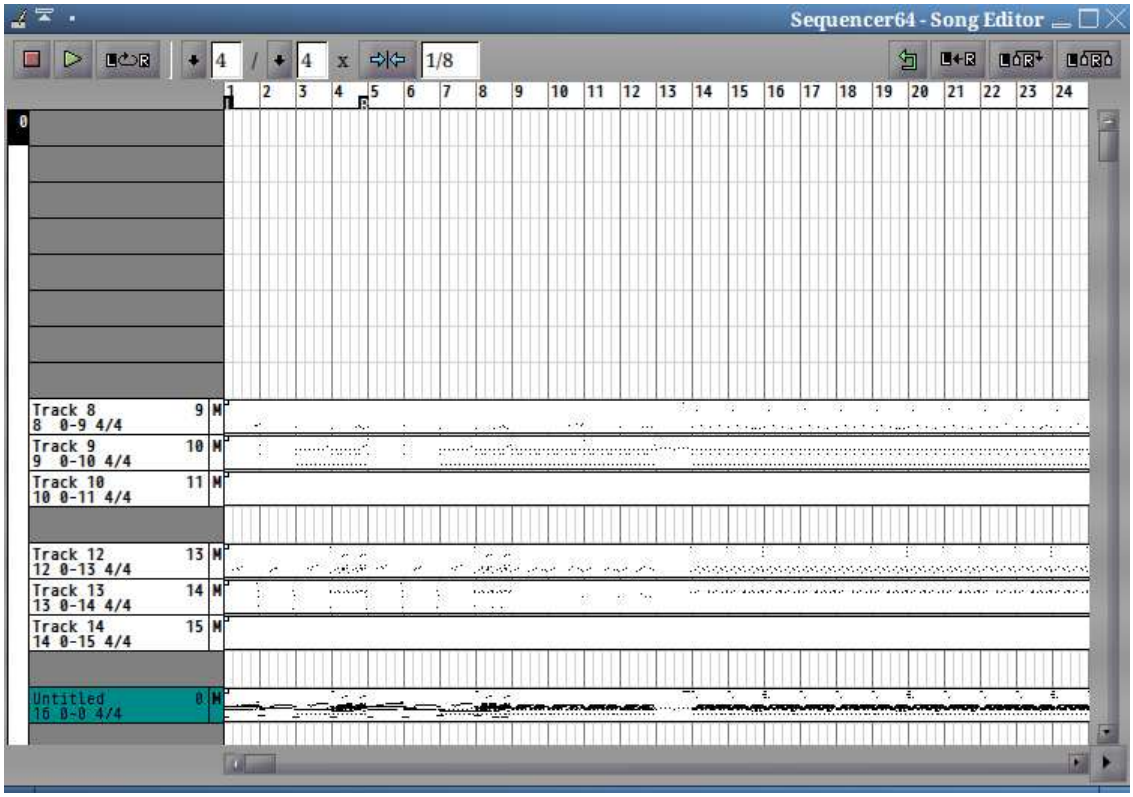


Figure 83: SMF 0 MIDI Song in the Song Editor

One is free to edit the imported tune to heart's content. Here, we added one instance of each track, including the SMF 0 track, to show what the imported song looks like.

13.2 Legacy Proprietary Track Format

The authors of *Seq24* took the trouble to make sure that the format of the MIDI files it write are compatible with other MIDI applications. *Seq24* also stores its own information (triggers, MIDI control information, etc) in the file, but marked so that other sequencers can read the file and ignore the *Seq24*-specific information.

Sequencer64 continues that MIDI-compliant behavior, but has improved the compliance just a little bit. We call that last chunk of sequencer-specific information the "proprietary track". Before we discuss that last, proprietary track, note that the normal MIDI tracks that precede it include the SeqSpec ("sequencer-specific", sort of) control tags shown in table 7 "SeqSpec Items in Normal Tracks" on page 145. These control tags are global constants in the *Seq24* source code, ranging from 0x24240001 to 0x24240013.

The `c_triggers` tag is obsolete.

New: The `c_musickey`, `c_musicscale`, and `c_backsequence` control tags are new with *Sequencer64* version 0.9.9.7 and above. They are now saved as additional information in each sequence in which they have been specified in the sequence editor. However, for backward compatibility (and because it is probably the more common use case), these items can also be saved globally for the whole MIDI song, as an option.

Table 7: SeqSpec Items in Normal Tracks

c_midibus	24 24 00 01 00 00 00 00
c_midich	24 24 00 02 00 00 00 00
c_triggers	24 24 00 04 00 00 00 00
c_timesig	24 24 00 06 00 00 00 00
c_triggers_new	24 24 00 08 00 00 00 00
c_musickey	24 24 00 11 00
c_musicscale	24 24 00 12 00
c_backsequence	24 24 00 13 00 00 00 00

Note that these tags (created by the application, but not present in the proprietary track, and perhaps also created by other MIDI applications) are preceded by the standard MIDI "FF 7F length" meta-event sequence.

The following discussion applies to the final "proprietary" track as saved in the legacy *Seq24* format.

After all the counted MIDI tracks are read, *Seq24* checks for extra data. If there is extra data, *Seq24* reads a long value. The first one encountered is a MIDI "sequencer-specific" (*SeqSpec*) section. It starts with

```
0x24240010 == a Seq24 c_midictrl proprietary value
```

Getting this value first is simplified MIDI in two ways. First, the second does not begin with any kind of track marker. MIDI requires an "MTrk" marker to start a track, though it also requires unknown markers to be supported. Some applications, like *timidity*, handle this situation. Others, like *midicvt*, complain about an unexpected header marker. Second, normally, MIDI wants to see the triad of

```
status = FF, type= 7F (proprietary), length = whatever
```

to precede proprietary data. Now, as shown by table 11 "Application Support for MIDI Files" on page 149, most applications accept the shortcut legacy format, but *midicvt* does not.

So, as a "bug" fix, we now write and read this information properly in *Sequencer64*; it is now preceded by the `0xFF 0x7F` marker.. We also need to be able to read legacy Seq24 MIDI files, so that ability has been preserved in *Sequencer64*..

Anyway, at this point, we have the **c_midictrl** information now. Next, we read a long value, seqs. It is 0.

```
24 24 00 10 00 00 00 00
```

Read the next long value, 0x24240003. This is **c_midiclocks**. We get a value of 0 for "TrackLength" (now a local variable called "busscount"):

```
24 24 00 03 00 00 00 00
```

If the buss-count was greater than 0, then for each value, we would read a byte value represent the bus a clock was on, and setting the clock value of the master MIDI buss. Another check for more data is made.

```
24 24 00 05 00 20 00 00
```

0x24240005 is **c_notes**. The value screen_sets is read (two bytes) and here is 0x20 = 32. For each screen-set:

```
len = read\_short()
```

If non-zero, each of the `len` bytes is appended as a string. Here, len is 0 for all 32 screensets, so the screen-set notepad is set to an empty string. Another check for more data is made.

```
24 24 00 07 00 00 00 78
```

0x24240007 is **c_bpmtag**. The long value is read and sets the perform object's bpm value. Here, it is 120 bpm. Another check for more data is made.

```
24 24 00 09 00 00 04 00
```

0x24240009 is **c_mutegroups**. The long value obtained here is 1024. If this value is not equal to the constant **c_gmute_tracks** (1024), a warning is emitted to the console, but processing continues anyway, 32 x 32 long values are read to select the given group-mute, and then set each of its 32 group-mute-states.

In our sample file, 32 groups are specified, but all 32 group-mute-state values for each are 0.

So, to summarize the legacy proprietary track's data, ignoring the data itself, which is mostly 0 values, as shown in table 8 "SeqSpec Items in Legacy Proprietary Track" on page 146

Table 8: SeqSpec Items in Legacy Proprietary Track

`c_midictrl`	`24 24 00 10 00 00 00 00`	
`c_midiclocks`	`24 24 00 03 00 00 00 00`	(buss count = 0)
`c_notes`	`24 24 00 05 00 20 00 00`	(screen sets = 32)
`c_bpmtag`	`24 24 00 07 00 00 00 78`	(bpm = 120)
`c_mutegroups`	`24 24 00 09 00 00 04 00`	(mg = 1024)

The new format (again, ignoring the data) takes up a few more bytes. It starts with the normal track marker and size data, followed by a made-up track name ("Sequencer64-S"), as shown in table 9 "SeqSpec Items in New Proprietary Track" on page 147.

For the new format, the components of the final proprietary track size are as shown here:

1. **Delta time**. 1 byte, always 0x00.
2. **Sequence number**. 5 bytes. OPTIONAL.
3. **Track name**. 3 + 10 or 3 + 15

Table 9: SeqSpec Items in New Proprietary Track

```
"MTrk" etc.   4d 54 72 6b 00 00 11 0d 00 ...
Track name    53 65 71 75 65 6e 63 65 72 32 34 2d 53
c_midictrl    ff 7f 04 24 24 00 10 00 (???)
c_midiclocks  ff 7f 04 24 24 00 03 00 (buss count = 0)
c_notes       ff 7f 46 24 24 00 05 00 20 00... (screen sets = 32)
c_bpmtag      ff 7f 08 24 24 00 07 00 00 00 78 (bpm = 120)
c_mutegroups  ff 7f a1 08 24 24 00 09 00 00 04 00... (mg = 1024)
```

4. **Series of proprietary specs**:
   - **Prop header**:
     - If legacy format, 4 bytes.
     - Otherwise, 2 bytes + varinum_size(length) + 4 bytes.
     - Length of the prop data.
5. **Track End**. 3 bytes.

13.3 MIDI Information

This section provides some useful, basic information about MIDI data.

13.3.1 MIDI Variable-Length Value

A variable-length value (VLV) is a quantity that uses additional bytes and continuation bits to encode large numbers without confusing a MIDI interpreter. See https://en.wikipedia.org/wiki/Variable-length_quantity.

The length of a variable length value obviously depends on the value it represents. Here is a simple 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
```

13.3.2 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.

Table 10: MIDI Meta Event Types

Type	Event
0x00	Sequence number
0x01	Text event
0x02	Copyright notice
0x03	Sequence or track name
0x04	Instrument name
0x05	Lyric text
0x06	Marker text
0x07	Cue point
0x20	MIDI channel prefix assignment
0x2F	End of track
0x51	Tempo setting
0x54	SMPTE offset
0x58	Time Signature
0x59	Key Signature
0x7F	Sequencer-Specific event

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 variable length value for 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.

13.3.3 MIDI Meta Events

Meta events are non-MIDI data of various sorts consisting of a fixed prefix, type indicator, a length field, and actual 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.

Timidity reads the legacy and new formats and plays the tune. *Sequencer64* saves the "b4uacuse" tune out, in both formats, with a "MIDI divisions" value of 192, versus its original value of 120. The song plays a little bit faster after this conversion.

The *midicvt* application does not read the legacy *Seq24* file format. It expects to see the MTrk marker. Even if the *midicvt* `--ignore` option is provided, *midicvt* does not like the legacy *Seq24* format, and ends with an error message. However, as shown by table 11 "Application Support for MIDI Files" on page 149, most applications are more forgiving, and can read (or ignore) the legacy format. The *gsequencer* application has some major issues in our installation, but it is probably our setup. (No JACK running?)

Table 11: Application Support for MIDI Files

Application	**Legacy**	**New**	**Original File**
ardour	TBD	TBD	TBD
composite	TBD	TBD	TBD
gsequencer	No	No	No
lmms	Yes	Yes	Yes
midi2ly	Yes	Yes	TBD
midicvt	No	Yes	Yes
midish	TBD	TBD	TBD
muse	TBD	TBD	TBD
playmidi	TBD	TBD	TBD
pmidi	TBD	TBD	TBD
qtractor	Yes	Yes	Yes
rosegarden	Yes	Yes	Yes
superlooper	TBD	TBD	TBD
timidity	Yes	Yes	Yes

13.4 More MIDI Information

This section goes into even more detail about the MIDI format, especially as it applies to the processing done by *Sequencer64*. The following sub-sections describe how *Sequencer64* parses a MIDI file.

13.4.1 MIDI File Header, MThd

The first thing in a MIDI file is The data of the header:

```
Header ID:     "MThd"         4 bytes
MThd length:     6            4 bytes
Format:        0, 1, 2        2 bytes
No. of track:  1 or more      2 bytes
PPQN:           192           2 bytes
```

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

13.4.2 MIDI Track, MTrk

The next part of the MIDI file consists of the tracks specified in the file. In SMF 1 format, each track is assumed to cover a different MIDI channel, but always the same MIDI buss. (The MIDI buss is not a data item in standard MIDI files, but it is a special data item in the sequencer-specific section of *Seq24/Sequencer64* MIDI files.) Each track is tagged by a standard chunk marker, "MTrk". Other markers are possible, and are to be ignored, if nothing else. Here are the values read at the beginning of a track:

```
Track ID:      "MTrk"         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.

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 passed along.

13.4.3 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, and the MIDI channel of the sequence is set.

13.4.4 Meta Events Revisited

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

```
Meta type:     varies         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 Seq24 "proprietary" events. These events are tagged with various values that mask off to 0x24240000. The parser reads the tag:

```
Prop tag:     0x242400nn      4 bytes
```

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

Anything 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.

13.5 Meta Events

Here, we summarize the MIDI meta events.

1. `FF 00 02 ssss`: Sequence Number.
2. `FF 01 len text`: Text Event.
3. `FF 02 len text`: Copyright Notice.
4. `FF 03 len text`: Sequence/Track Name.
5. `FF 04 len text`: Instrument Name.
6. `FF 05 len text`: Lyric.
7. `FF 06 len text`: Marker.
8. `FF 07 len text`: Cue Point.
9. `FF 08 through 0F len text`: Other kinds of text events.
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

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

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

13.5.1 Sequence Number (0x00)

```
FF 00 02 ss ss
```

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

13.5.2 Track/Sequence Name (0x03)

```
FF 03 len text
```

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

13.5.3 End of Track (0x2F)

```
FF 2F 00
```

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

13.5.4 Set Tempo Event (0x51)

The MIDI Set Tempo meta event sets the tempo of a MIDI sequence in terms of the microseconds per quarter note. This is a meta message, so this event is never sent over MIDI ports to a MIDI device. After the delta time, this event consists of six bytes of data:

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

Example:

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

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

Since there are 60,000,000 microseconds per minute, the event above translates to: set the tempo to 60,000,000 / 500,000 = 120 quarter notes per minute (120 beats per minute).

This 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.

13.5.5 Time Signature Event (0x58)

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

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

The time signature is expressed as four numbers. `nn` and `dd` represent the numerator and denominator of the time signature as it would be notated. The 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 *Sequencer64*, the `c_timesig` SeqSpec event is given priority. The conventional time signature is used only if the `c_timesig` SeqSpec is not present in the file.

13.5.6 SysEx Event (0xF0)

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

Sequencer64 has some code in place to store these messages, but the data is currently not actually stored or used. Although there is some infrastructure to support storing the SysEx event within a sequence, the SysEx information is simply skipped. *Sequencer64* warns if the terminating 0xF7 SysEx terminator is not found at the expected length. Also, some malformed SysEx events have been encountered, and those are detected and skipped as well.

13.5.7 Sequencer Specific (0x7F)

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

```
FF 7F len data
```

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

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

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

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

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

13.5.8 Non-Specific End of Sequence

Any other statuses are deemed unsupportable in *Sequencer64*, and abort parsing with an error.

If the –bus option is in force, 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, *Sequencer64* adds the sequence to the encoded tune.

14 Kudos

This section gives some credit where credit is due. We have contributors to acknowledge:

- *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.
- *Stan Preston (stazed)*: ideas for some upcoming improvements based on his *seq32* project. A lot of ideas. And a lot of code!

Also some bug-reporters and testers:

- *gimmeapill*: testing, bug-reports, and, um, "marketing".
- *muranyia*: feature request for numbered piano keys and bug-reports.
- *F0rth*: a request for scripting support, a possible future feature.
- *triss*: a request for OSC support, a possible future feature.
- *ssj71*: a request for an LV2 plugin version, a possible future feature.

There are a number of authors of *Seq24*. ideas from other *Seq24* fans), and some deep history, as one can see in figure 25 "Help Credits" on page 36, and in figure 26 "Help Documentation" on page 37. All of these authors, and more, have contributed to *Sequencer64*, whether they know it or not. The original author is Rob C. Buse; where the word "I" occurs, that is probably him. Without his work, we would never have started *Sequencer64*.

From the original author:

> *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.

This project deserves to stay alive! (And it is alive! A new version, 0.9.3, has come out from the LaunchPad group! It corrects a problem with MIDI Clock drift). 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 a few features we've found useful, to make it the "vi of sequencers".

Always remember that, without *Seq24* and its authors, *Sequencer64* would never have come into being.

15 Sequencer64 JACK Support

This section describes some details concerning the JACK support of *Sequencer64*. As with *Seq24*, *Sequencer64* has JACK transport support. But, if one wants to use the older version of *Sequencer64* (versions 0.9.x) with JACK MIDI, one needs to expose the ALSA ports to JACK using `a2jmidid --export-hw` and connect the resultant MIDI JACK ports oneself, using *QJackCtl*, for example.

To enable the JACK transport support at run-time, the options `-j`/`--jack-transport`, `-J`/`--jack-master`, and `-C`/`--jack-master-cond` are available.

With version 0.90, *Sequencer64* can be built to support the legacy ALSA interface, the PortMIDI interface, or, best of all, the native JACK MIDI interface, loosely based on the RtMIDI project (see [13]). This mode also supports fallback-to-ALSA if the JACK server is not running.

- *Sequencer64*. . This application is built when the `--enable-alsamidi` option is specified at "configure" time. It is basically the 0.9.x version of *Sequencer64* application. However, this build is no longer the default.
- *seq64*. . This application is built when the `--enable-rtmidi` option is specified at "configure" time. This build is now the default build.
- *seq64portmidi*. . This application is built when the `--enable-portmidi` option is specified at "configure" time. This build is *deprecated*. It works with Linux, but, for Windows support, we will instead add Windows API calls to the "rtmidi" build of the project. We won't discuss this version at all. We've tested it for playback, but nothing else.

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

15.1 Sequencer64 JACK Transport

This section is just underway. Here are some of the topics to be discussed:

1. What JACK functions are supported for JACK Transport.
2. Exposing the ALSA MIDI ports to JACK, when using the legacy ALSA version of *Sequencer64*.
3. Fixes to JACK Master mode.
4. Interactions with the Klick and Hydrogen applications.
5. Patches from the new (!) version of Seq24, 0.9.3, to correct for MIDI Clock drift.

In the meantime, the text files in the project's `contrib/notes` directory provide some useful setup notes.

First, if *Sequencer64* is run in JACK mode without JACK running on the system, it will take awhile to come up (in ALSA mode). If run from the console, one will see:

```
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jackdmp 1.9.11
Copyright 2001-2005 Paul Davis and others.
Copyright 2004-2014 Grame.
jackdmp comes with ABSOLUTELY NO WARRANTY
This is free software, and you are welcome to redistribute it
under certain conditions; see the file COPYING for details
```

```
JACK server starting in realtime mode with priority 10
self-connect-mode is "Don't restrict self connect requests"
audio_reservation_init
Acquire audio card Audio1
creating alsa driver ... hw:PCH|hw:PCH|1024|3|48000|0|0|nomon|swmeter|-|32bit
ATTENTION: The playback device "hw:PCH" is already in use. Please stop the
application using it and run JACK again
JackTemporaryException : now quits...
Cannot initialize driver
JackServer::Open failed with -1
Failed to open server
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
 . . .
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for 4294967295,
skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for 4294967295,
skipping unlock
[JACK server not running, JACK sync disabled]
```

MORE TO COME.

15.2 Sequencer64 Native JACK MIDI

This section discusses the new *seq64* application, which supports native JACK MIDI. It is now the *official* version of *Sequencer64*, and new bugs will be fixed mainly in this version.

The first thing to note about *seq64* is that it supports both ALSA and JACK MIDI. If one runs it to support JACK, and JACK is not present, then *seq64* falls back to ALSA support.

To run seq64 to support JACK, for now, one must add the `-t` or `--jack-midi` option. Why `-t`? We are running out of option letters. And eventually we will make the `-t` option the default.

The next thing to note about *seq64* is that the original JACK-transport options are still present, but the JACK transport support is *separate* from the JACK MIDI support. The JACK transport client is an invisible client with the name "seq64-transport", while the JACK MIDI client is visible in *QJackCtl*, and the ports created are part of the "seq64" client.

To enable the JACK MIDI support, the `-t`/`--jack-midi` options are available. When enabled, the "transport" button in the main window will show the word "Native" (even when JACK transport is also enabled). Please note that, if you select the jack-midi option and JACK is not available, *seq64* will start in ALSA mode and *it will save* that mode when exiting.

The JACK options are sticky options. That is, they are saved to the "rc" configuration file, so one does not have to specify them in subsequent *seq64* sessions.

15.2.1 Sequencer64 JACK MIDI Output

By default (or depending on the "rc" configuration file), the new `seq64` version of *Sequencer64* will automatically connect the ports that it finds to `seq64`, as shown in the following figure:

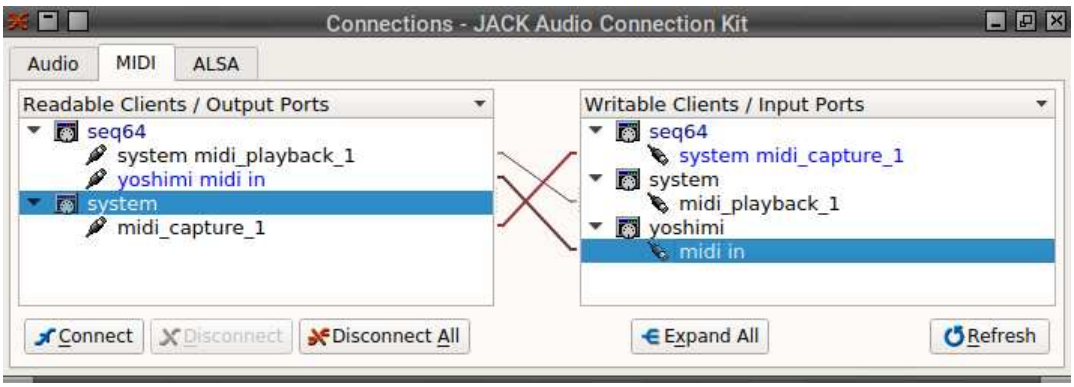


Figure 84: JACK MIDI Auto-Connect

The `seq64:system midi_playback_1` output port shown in the left panel is created by `seq64`. It connects it to the `system:midi_playback_1` port in the right panel, which is actually the input for the *Korg nanoKEY2* controller. ALSA detects the real name of this device, but JACK does not. Note that this connection is not useful unless `seq64` could send setup information to the *nanoKEY2*, which is probably not the case (as far as we know).

More useful is the automatic connection between `seq64:yoshimi midi in` in the left (output) panel and `yoshimi:midi in` in the right (input) panel. With it it, *Sequencer64* patterns with the proper output-buss setting can play to the *Yoshimi* software synthesizer.

Note that the index, client, and buss numbers are all the same. There's actually a bug here, since all `seq64` ports should have the same client number. However, in JACK, clients and ports are referred to by name, not number, and so functionality is not affected.

Entry 0 is, as already noted, not useful unless the *nanoKEY2* can accept input control. Entry 1 allows `seq64` to send MIDI to *Yoshimi*. A pattern must specify output buss 1 in order for the MIDI to readh *Yoshimi*. Another option, normally for testing only, is to specify the "bus" option on the command line:

```
$ seq64 --jack-midi --bus 1
```

With that option, all patterns send to buss 1.

15.2.2 Sequencer64 JACK MIDI Input

One more connection to note is the input connection to `seq64`. Referring back to figure 84 "JACK MIDI Auto-Connect" on page 158we see that `system:midi_capture_1` in the left (output) panel is connected to `seq64:system midi_capture_1` in the right (input) panel. This allows the *nanoKEY2* MIDI output port to feed `seq64`, which can then record the input notes, and also forward them to *Yoshimi* so that they can be heard.

This input port is also shown in the **File / Options / MIDI Input** tab, shown here:

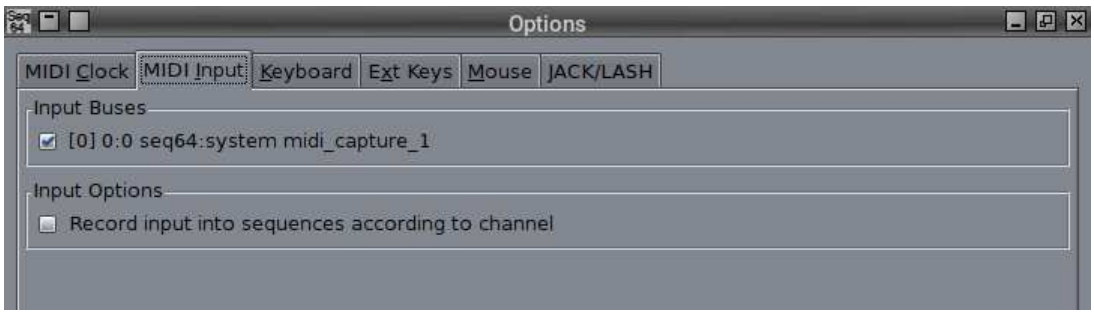


Figure 85: JACK MIDI Input Ports

When the check-box for that buss is selected, the input can be captured by `seq64`.

15.2.3 Sequencer64 JACK MIDI Virtual Ports

The manual-versus-normal port support for JACK MIDI is essentially the same as that for ALSA. Currently, the same option name is used (we will provide a more generic option-name soon). The `-m`/`--manual-alsa-ports` option actually provides what are known as "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 *seq64* to automatically connect to all existing JACK MIDI ports, *do not* use the `-m`/`--manual-alsa-ports` option... use the `-a`/`--auto-alsa-ports` option. Both options apply to both ALSA and JACK, but we do not want to change the options names yet.

If one wants the freedom to make the connections oneself, or with a session manager, then use the manual/virtual option.

Here are the ports created in manual/virtual mode:

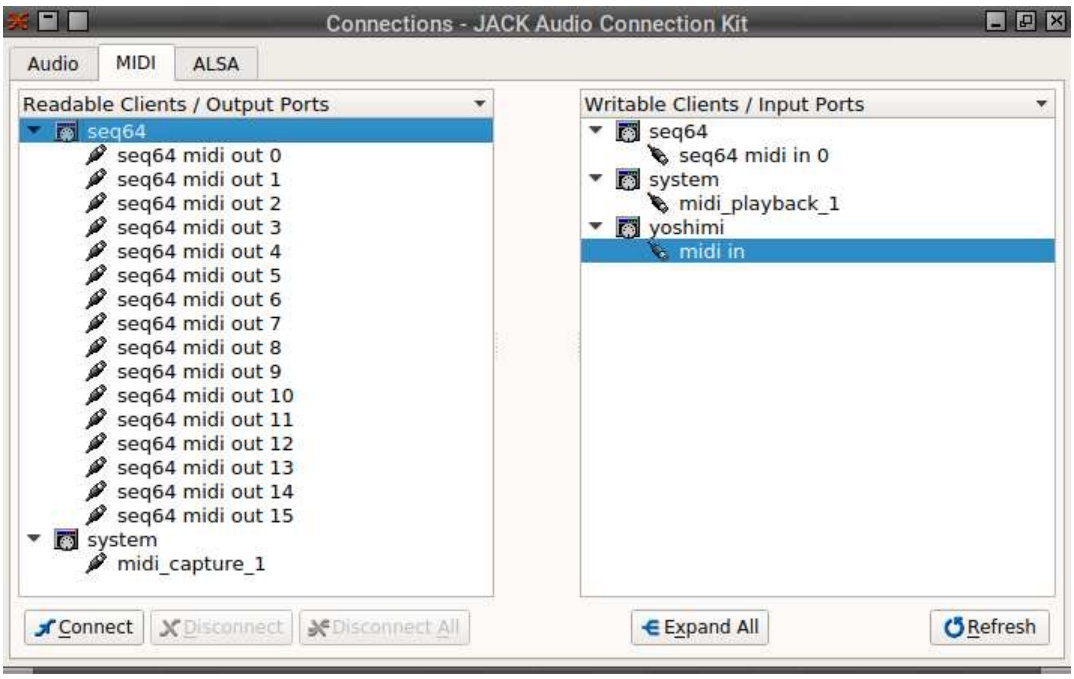


Figure 86: JACK MIDI Manual Ports

One sees that `seq64` creates 16 output ports (busses), and one input port (buss). One also sees that `seq64` *does not* connect the ports automatically. The user or the session manager will have to make those connections.

The **MIDI Clock** and **MIDI Input** tabs reflect in an obvious manner what is seen in *QJackCtl*, so we won't bother to show those tabs.

15.2.4 Sequencer64 JACK MIDI and a2jmidid

One more thing to show is that `seq64` can deal with the odd naming of JACK ports created by the *a2jmidid* application.

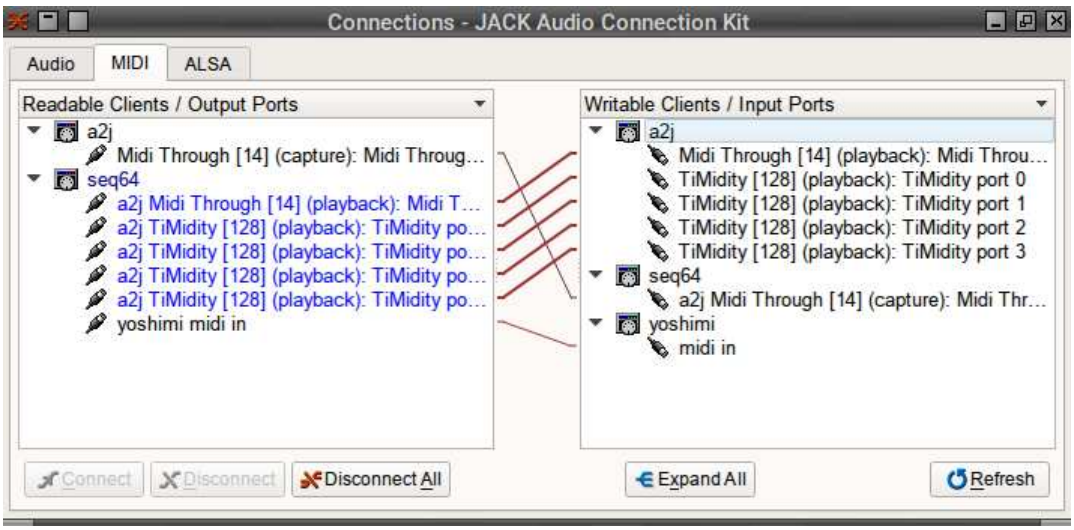


Figure 87: JACK MIDI a2jmidid Ports

One can see in the right (input) panel that that the `a2j` client creates 5 entries, one for "Midi Through", and four for the *TiMidity* client. In the left (output) panel, one sees (in blue) the output ports that `seq64` creates to connect to the ports created by *a2jmidid*.

Also note the true JACK output port, `seq64:yoshimi midi in` to connect to the input port `yoshimi:midi in`.

Again, if these automatic connections get in the way, run `seq64` in manual/virtual mode.

When recording, do not forget the step option. If one paints notes with the mouse, the note is previewed, and the note position advances with each click. If one paints notes via an external MIDI keyboard, the notes are painted and advanced, but they are not previewed. To preview them, click the "pass MIDI in to output" button in the pattern editor window to activate so that they will be passed to your sound generator.

16 Summary

In summary, we can say that you will find *Sequencer64* intriguing.

Contact: If you have ideas about *Sequencer64* 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.

17 References

The *Sequencer64* 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] amSynth team, Nick Dowell *amSynth and Demos with Calf Effects.* http://amsynth.com/amsynth.html Includes links to demos and the source code. 2015.

[3] Bristol team Nick Copeland *Bristol: A Vintage Synthesizer Emulator* http://www.linuxsynths.com/BristolPatchesDemos/bristol.html 2014.

[4] FluidSynth team *FluidSynth: A SoundFont Synthesizer* http://www.fluidsynth.org/ 2014.

[5] Jay Capela Music *"Combine": A Seq24 Demonstration* https://www.youtube.com/watch?v=fUiXbVT0bJQ 2010.

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

[7] LinuxSynths team, briandc@linuxsynths.com *A Sonic Palette on the Linux Platform.* http://www.linuxsynths.com/ 2015.

[8] Dave Phillips *At the Sounding Edge: Introducing seq24.* http://www.linuxjournal.com/article/8304 Linux Journal, May 12, 2005.

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

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

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

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

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

[14] Seq24 Team. *The home site for the Sequencer64 looping sequencer.* http://www.filter24.org/seq24/download.html 2010.

[15] pneumanlsd. *Linux audio demo: Live sequencing with seq24* https://www.youtube.com/watch?v=f8zLV0vlSpY 2010.

[16] synthWF. *Misty Corridor - Seq24 (with QSynth)* https://www.youtube.com/watch?v=RH99zHvffGQ 2012.

[17] pneumanlsd. *Linux music tutorial: seq24, part 1* https://www.youtube.com/watch?v=J2WDHS1wYeM 2010.

[18] pneumanlsd. *Linux music tutorial: seq24, part 2* https://www.youtube.com/watch?v=i3Vpi3oxdqk 2010.

[19] Seq24 Team. *The home site for the Sequencer64 looping sequencer.* https://launchpad.net/seq24 2016.

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

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

[22] sbrauer. *The home site for the original Seq42 looping sequencer.* https://github.com/sbrauer/seq42 2016.

[23] Stan Preston (stazed). *A fork of the Seq42 looping sequencer.* https://github.com/Stazed/seq42 2016.

[24] Chris Ahlstrom. *A reboot of the Seq24 project as "Sequencer64".* https://github.com/ahlstromcj/sequencer64/ 2015-2017.

[25] Chris Ahlstrom. *The Sequencer64 User Manual.* https://github.com/ahlstromcj/sequencer64-doc/ 2015-2017.

[26] Chris Ahlstrom. *Sequencer64 Debian Packages.* https://github.com/ahlstromcj/sequencer64-packages/ 2017.

[27] Kevin at subatomicglue.com *Subatomic Mods for Seq24 Win32* http://www.subatomicglue.com/seq24/ 2010.

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

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

[30] The Wootangent man. *Sequencer64 Tutorial Video, Part 1.* http://wootangent.net/2010/10/linux-music-tutorial-seq24-part-1/ 2010.

[31] The Wootangent man. *Sequencer64 Tutorial Video, Part 2.* http://wootangent.net/2010/10/linux-music-tutorial-seq24-part-2/ 2010.

[32] Yoshimi team abrolag@users.sourceforge.net *The download site for the Yoshimi software synthesizer.* http://yoshimi.sourceforge.net/ 2015.

[33] Yoshimi team *The alternate location for the Yoshimi source-code.* https://github.com/abrolag/yoshimi/ 2015.

[34] Chris Ahlstrom *A Yoshimi User Manual.* https://github.com/ahlstromcj/yoshimi-doc/ 2015.

[35] Chris Ahlstrom *A Yoshimi Cookbook.* https://github.com/ahlstromcj/yoshimi-cookbook/ 2015.

[36] Mark McCurry, Paul Nasca (ZynAddSubFX team) *The download site for the ZynAddSubFX software synthesizer.* http://zynaddsubfx.sourceforge.net/ 2015.

Index