Sequencer64 User Manual

Chris Ahlstrom
(ahlstromcj@gmail.com)

November 2, 2015

Contents

List of Figures

List of Tables

1 Introduction

This document describes how to use *Sequencer64* [11], through version 0.9.9.6, dated approximately November, 2015.

1.1 Sequencer64: What?

Sequencer64 is an continuation of *Sequencer24*, which itself is a continuation 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 now a necessity.

Seq24 was, a few years back, a very active project, with a number of contributors, who have created patches, additional functionality, and even ports to Windows. We have been assiduous about searching for all of these updates, and incorporating them into *Sequencer64* where feasible. There are still some things to be done, including a port to Windows/Portmidi (but that is a low priority).

Sequencer64 is not a synthesizer. It requires a hardware synthesizer, or a software synthesizer such as Timidity [13], FluidSynth [4], ZynAddSubYX [20], Yoshimi [16] [17], AmSynth [2], Bristol [3], and others (see [6] for a fairly comprehensive list of "Linux" synthesizers).

Sequencer64, like *Seq24*, is meant to work 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.

There are a number of authors of *Seq24*, and currently only one author of *Sequencer64*, as one can see in figure 15 ("Help Credits") on page 23, and in figure 16 ("Help Documentation") on page 24. All of these authors 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.

Now, unfortunately, *Seq24* is not under active development (2010 seems to be the last update). There are a number of minor 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 is consolidation of all the features and fixes that *Seq24* has accumulated in various forks over the years.

Also, although "feature-complete", *Seq24* could use a few more features, such as "infinite patterns" (i.e. "tracks"), better annunciation, uniformity, ease-of-use of keystrokes, ways to work around the need for two mouse buttons, a more up-to-date user-interface framework, a few bug fixes, fix the formatting of the output "rc" configuration file, beef up the support for the "user" configuration file, and a way to edit parts of the MIDI file textually.

> *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! Taking advantage of Rob's generosity, we've created a continuation, a refactoring, and an improvement (we hope) of *Seq24*. It preserves the lean nature of *Seq24*, and might even make it a bit leaner, while adding a few features we've found useful. 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*:

- Additional mouse and keystroke support in the pattern and song editor windows:
  - Ability to move selected notes or triggers using the arrow keys.
  - Better explanation of existing keystroke support for that window.
  - Additional keystroke support found by Googling for seq24 all around the Internet.
  - Ability to use the Mod4/Super/Windows key to keep editing (painting) mode enabled after releasing the right mouse button, for some of the modern crappy touchpads shipped with laptops.

- The "user" configuration is now written to disk, and we will be adding settings to it in the near future.
- 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.
- Small improvements in appearance:
  - Support for showing empty (i.e. having only meta events) sequences in a highlighting color.
  - 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 build time, that is bolder and has a more modern, anti-aliased look.
- 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 file.
- More musical scales (harmonic minor, melodic minor, and whole tone scales) have been added.
- 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 type of MIDI event container used, which greatly speeds up the loading of a MIDI file, especially in debug mode. If the new container reduces the maximum output of the sequencer, we will dump the slow container into a vector container for faster playback.
  - The code was reformmated using *astyle* and personal preferences. Modules were split into smaller units. Header file "includes" were moved into the "cpp" files to reduce header-file dependencies and build times, and to facilitate modularity and understanding of the code.
  - 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 windows) is now locked, since enlarging it offers no benefits.
  - Support for LASH is now a run-time option (as well as a build option).
  - Support for reading and writing configuration files from the user's `$HOME/.config/sequencer64` directory.

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.

1.4 Document Structure

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

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 ALSA 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:

```
$ sequencer64 --bus 5 b4uacuse-seq24.midi &
```

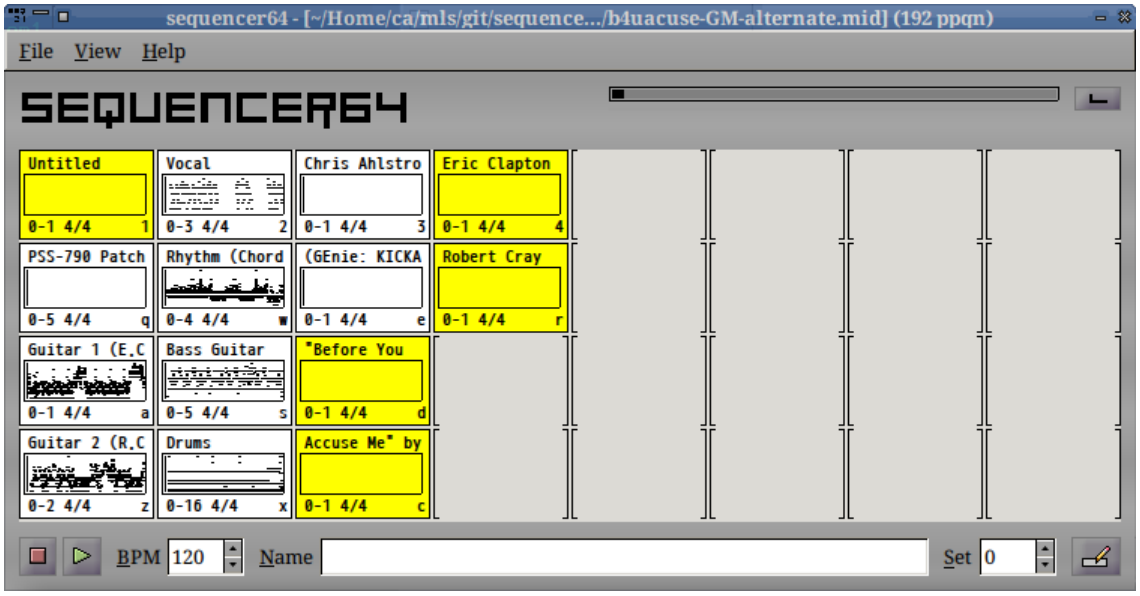

Figure 1: Sequencer64 Main Screen

Then the *Sequencer64* main window appears, as shown in figure 1 ("Sequencer64 Main Screen") on page 7. It has a two minor differences from the *Seq24* main window: the highlighting of empty patterns in yellow; and a different font.

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

2 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 (e.g. "sequence", "pattern", "loop", and "slot"), so it is good to clarify the usage here.

2.1 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.

2.1.1 Concepts / Terms / armed

An armed sequence is a sequence 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 on the 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.

2.1.2 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.

2.1.3 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 that can toggle 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".

2.1.4 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.

2.1.5 Concepts / Terms / measures ruler

The *measures ruler* is the bar at the top of the Song Editor arrangement window that shows the numbering of the measures in the song. Left and right 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*.

2.1.6 Concepts / Terms / muted

The opposite of section 2.1.1 ("Concepts / Terms / armed") on page 8.

2.1.7 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).

2.1.8 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.

2.1.9 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".

2.1.10 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.

2.1.11 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.

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.

2.1.12 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.

2.1.13 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.

2.1.14 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.

2.1.15 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.)

2.1.16 Concepts / Terms / song

A *song* is a collection of patterns in a specific layout, as assembled via the Song Editor window. Also see 2.1.9

2.2 Concepts / Sound Subsystems

2.2.1 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].

2.2.2 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 [9] 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 [12]) 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".)

2.2.3 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 [5].

3 Menu

The *Sequencer64* menu, as seen at the top of figure 1 ("Sequencer64 Main Screen") on page 7, is fairly simple, but it is important to understand the structure of the menu entries.

3.1 Menu / File

The **File** menu is used to save and load standard MIDI files. *Sequencer64* should be able to handle any Format 1 standard files that any other sequencer is capable of exporting.

The *Sequencer64* menu entry contains the sub-items shown in figure 2 ("Sequencer64 File Menu Items") on page 11. The next few sub-sections discuss the sub-items in the *File* sub-menu.

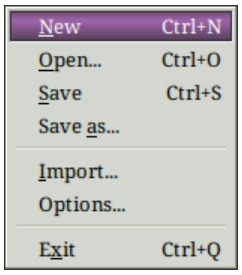

Figure 2: Sequencer64 File Menu Items

1. **New**
2. **Open...**
3. **Save**
4. **Save As...**
5. **Import...**

6. **Options...**
7. **Exit**

3.2 Menu / File / New

The **New** menu entry clears out any current song and patterns, allowing one to create news ones from scratch. If unsaved changes are pending, the user will be prompted to save the changes.

Currently, the detection of situations requiring a save (or not requiring a save) needs a bit of work.

3.2.1 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:

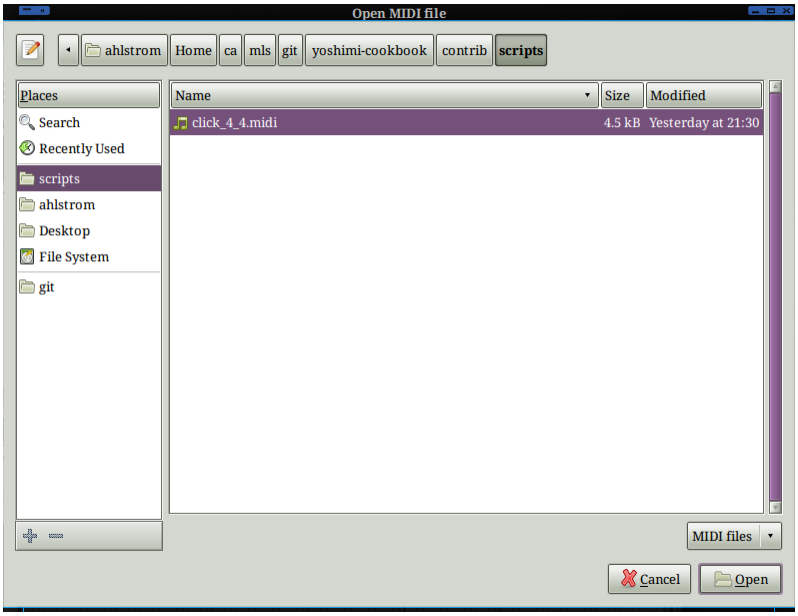

Figure 3: File Open

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!

3.2.2 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 GTK+2 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 GTK+2 file dialog:

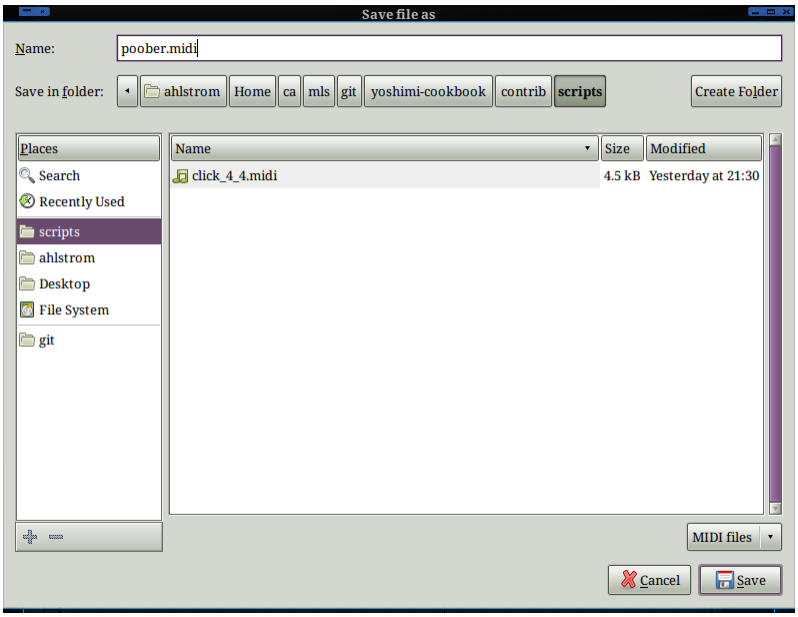

Figure 4: File Save As

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 subsections, as discussed in section 6.2.2 ("Song Editor / Arrangement Panel / Piano Roll") on page 48, have been saved. (But the L and R marker positions are not saved.)

Compare the sizes of the original project MIDI file, `contrib/b4uacuse.mid`, and the output MIDI file after *Sequencer64* saved the patterns and the song layout we created, `contrib/b4uacuse-seq24.midi`. The latter is a lot bigger.

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*. Unfortunately, this format is not quite standard, and a few MIDI applications may produce error messages (as opposed to just ignoring it) when parsing this section.

New: Therefore, in its normal mode, *Sequencer64* saves this information in a more MIDI-compliant format, 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 10.1 ("Legacy Proprietary Track Format") on page 63.

3.2.3 Menu / File / Import

The **Import** menu entry allows one to import a Format 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.

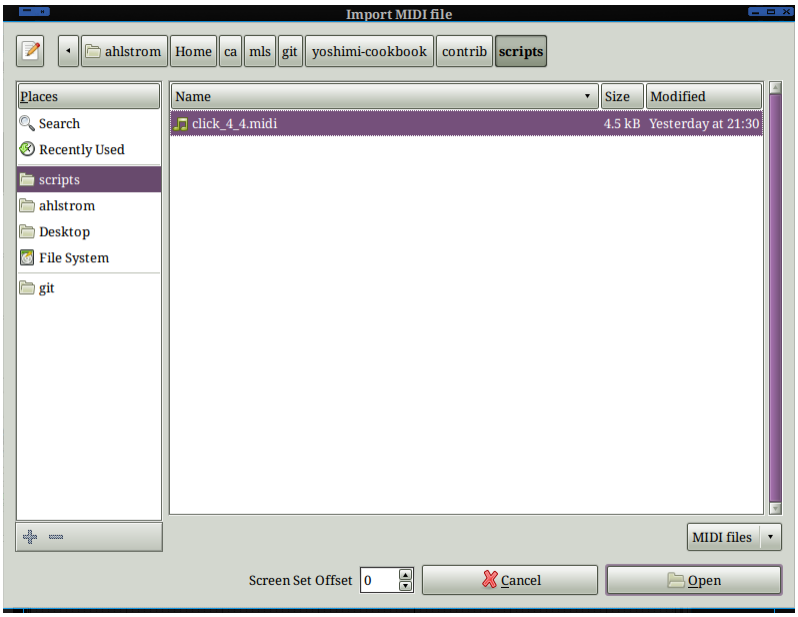

Figure 5: File Import

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, as shown in the following diagram, which shows an import of the `contrib/b4uacuse.mid` file, which contains a transcription of an Eric Clapton tune that we'd made over 20 years ago and had uploaded to the *GEnie* network service.

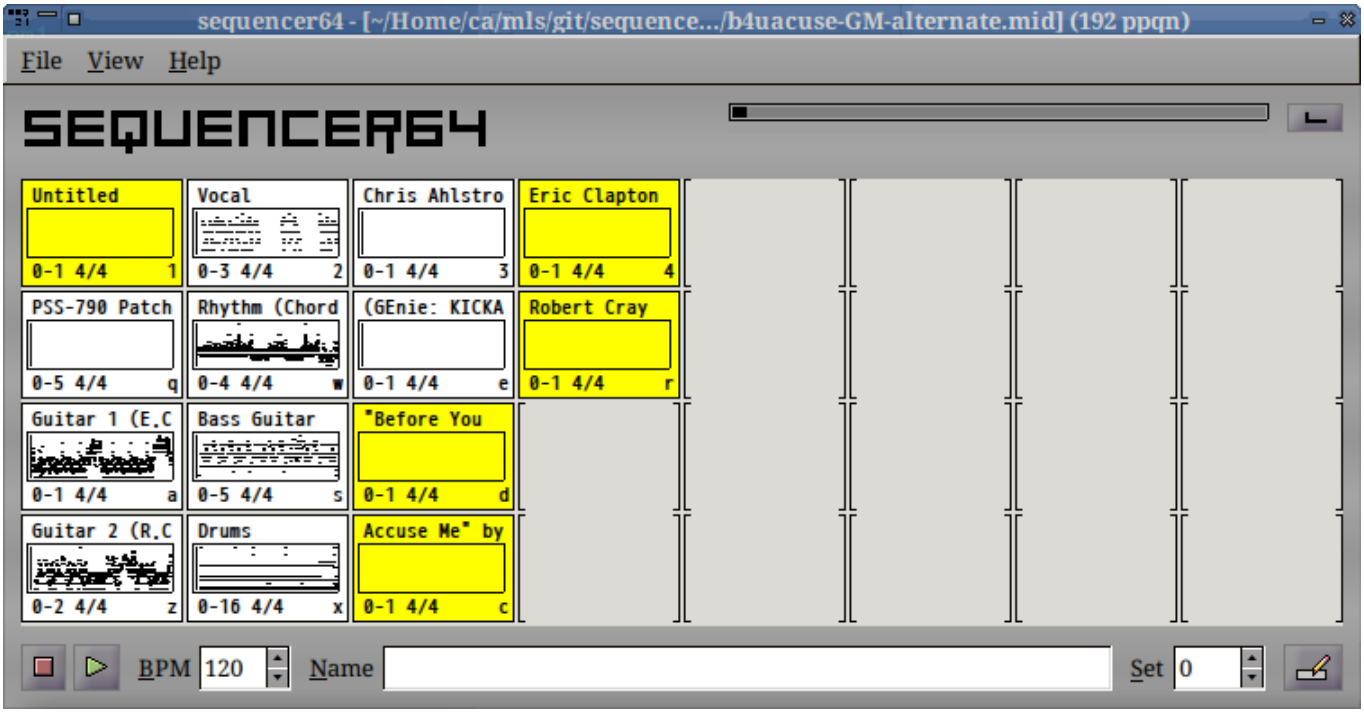

Figure 6: Imported MIDI Song

This song has a number of non-event tracks containing labels. **New:** These tracks are treated as "empty",

and show up with a yellow background. They are ignored on playback. Some of the tracks which look empty, but are not yellow, contain program-change events.

Unfortunately, this song was created before the days of General MIDI. It was scored for the Yamaha PSS-790 consumer-level synthesizer, definitely not a GM-compliant device. One can use the MIDI-conversion project, *midicvt* (see reference [8]), to convert it to General MIDI format, including General MIDI drums.

3.2.4 Menu / File / Options

The **Options** menu item provides a number of settings in one tabbed dialog, shown in the figure below. 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.

3.2.4.1 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 one can play music with. The items that appear in this tab depend on three 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 (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 `sequencer64rc` configuration file, as described in section 7.7 ("Sequencer64 / Other Sections") on page 57

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. 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    USB2.0-MIDI                      USB2.0-MIDI MIDI 1
 24:1    USB2.0-MIDI                      USB2.0-MIDI MIDI 2
128:0    TiMidity                         TiMidity port 0
128:1    TiMidity                         TiMidity port 1
128:2    TiMidity                         TiMidity port 2
128:3    TiMidity                         TiMidity port 3
130:16   seq24                            seq24 in
```

For some reason, the *Yoshimi* input port is not showing up in the output of `aplaymidi`, though, as shown in figure 8 ("MIDI Clock, Manual ALSA Option Off") on page 17, *Sequencer64* sees it on port 7. Perhaps that application is not providing a good ALSA device name. 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.

The following figure shows the result with the manual ALSA option of *Sequencer64* turned on:

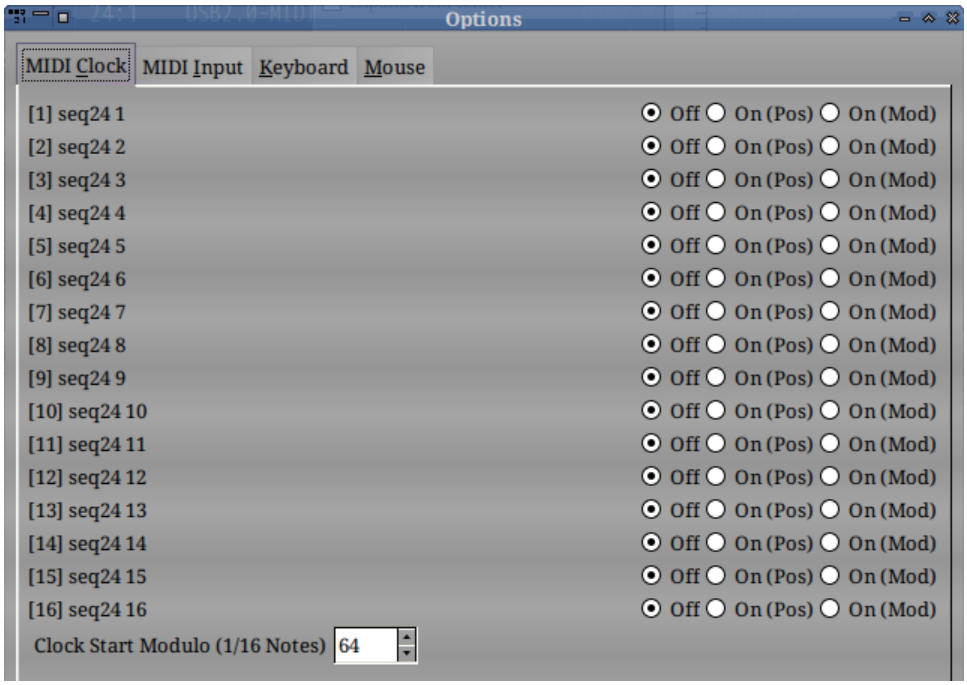

Figure 7: MIDI Clock, Manual ALSA Option On

It basically shows the 16 MIDI output busses that *Sequencer64* can drive. One would have to use an 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.

The following elements are present in this dialog:

1. **Buss Name**
2. **Off**
3. **On (Pos)**
4. **On (Mod)**
5. **Clock Start Modulo**

1. Buss Name. These labels indicate the output busses (ports) of *Sequencer64*. They range from **[1] seq24 1** to **[16] seq24 16**.

2. 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* with MIDI data, we found that this setting is the one that must be made in order for *Yoshimi* to produce a sound.

3. 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.

4. 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.

5. 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 7.7 ("Sequencer64 / Other Sections") on page 57.

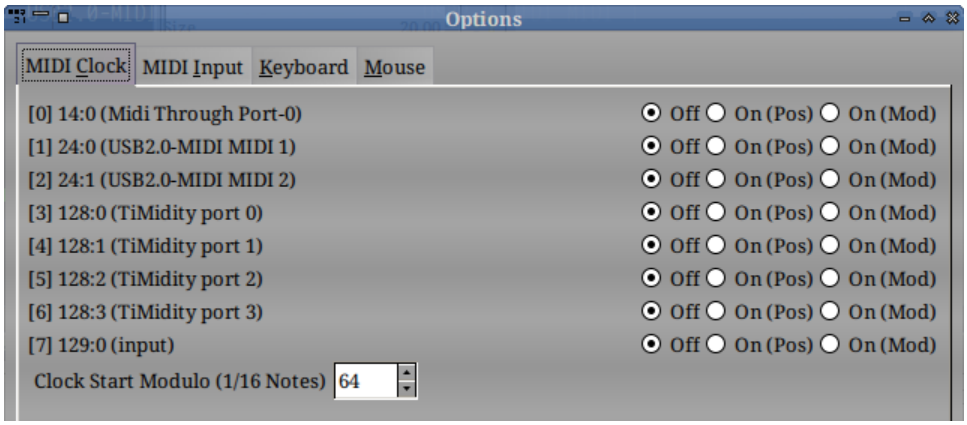

Figure 8: MIDI Clock, Manual ALSA Option Off

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 two MIDI ports on the USB cable, the four ports provided by *Timidity*, and the unlabelled port provided by *Yoshimi*.

One could theoretically play music through 6 or 7 devices using *Sequencer64* with this setup.

3.2.4.2 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
130:0    seq24                            [1] seq24 1
130:1    seq24                            [2] seq24 2
130:2    seq24                            [3] seq24 3
 . . .   . . .                               . . .
130:15   seq24                            [16] seq24 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 (see below) is turned on, then the only item in the **MIDI Input** tab is the single MIDI input buss provided by *Sequencer64*: **[0] seq24 0**, or, since the MIDI Thru port takes slot 0, **[1] seq24 1**.

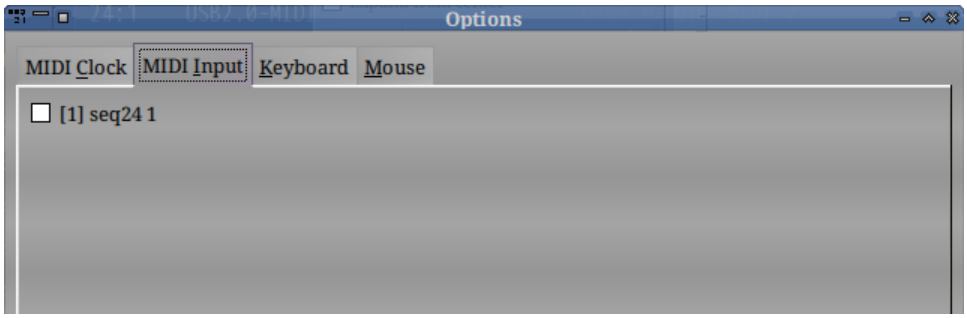

Figure 9: MIDI Input, Manual ALSA Ports On

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 5.4 ("Pattern Editor / Bottom Panel") on page 42.)

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

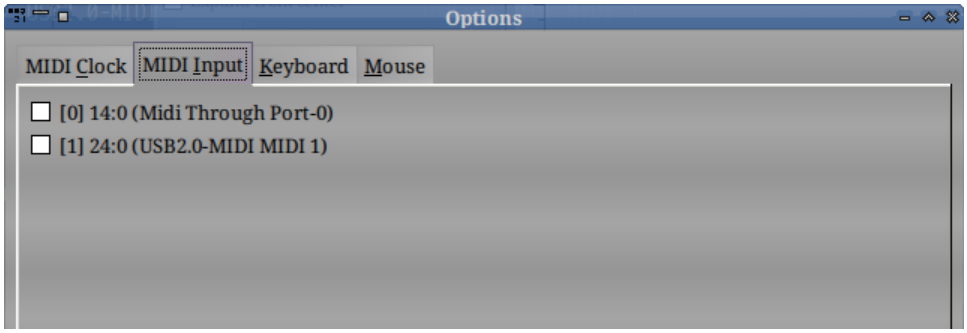

Figure 10: MIDI Input, Manual ALSA Ports Off

For example, one could check input #1 to have *Sequencer64* record MIDI from an old-fashioned MIDI keyboard that is connected to the USB MIDI cable. 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 **??** ("**??**") on page **??**.

3.2.4.3 Menu / File / Options / Keyboard

Sequencer64, as befits a good application, allows extensive use of keyboard shortcuts to make operations go faster than when using a mouse. The **Keyboard** tab allows for the configuration of these keyboard shortcuts.

Warning: 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!

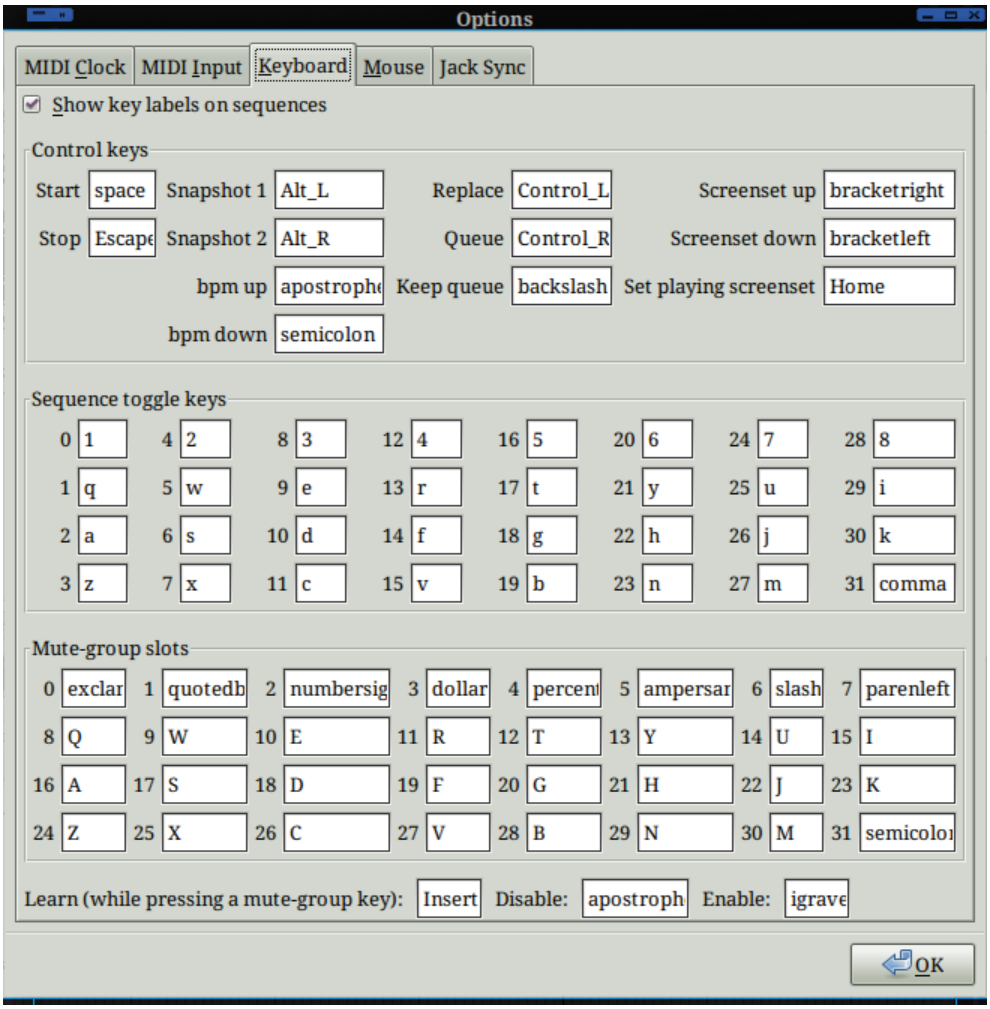

Figure 11: File / Options / Keyboard

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.

1. **Show key labels on sequences**
2. **Control keys**
3. **Sequence toggle keys**
4. **Mute-group slots**
5. **Learn**
6. **Disable**
7. **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. This feature is useful for live playback and control of a song.

2. Control keys. This block of fields provides shortcut keys for many operations of *Sequencer64*.

1. **Start**. Key: **space**.
2. **Stop**. Key: **Escape**.
3. **Snapshot 1**. Key: **Alt_L**.

4. **Snapshot 2**. Key: **Alt_R**.
5. **bpm up**. Key: **apostrophe**.
6. **bpm down**. Key: **semicolon**.
7. **Replace**. Key: **Control_L**.
8. **Queue**. Key: **Control_R**.
9. **Keep queue**. Key: **backslash**.
10. **Screenset down**. Key: **bracketleft**.
11. **Screenset up**. Key: **bracketright**.
12. **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.

The "keep queue" functionality allows the queue to be held without holding down a button the whole time.

3. 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!

4. 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!

Apparently groups work with the playing screen set only. Change the screenset and give the command to make it the playing one (e.g. set the HOME key for this purpose.)

5. 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.

6. Disable. TODO: What gets disabled? It is the **apostrophe** key by default.

7. Enable. TODO: What gets enabled? It is the **igrave** (back-tick) key by default.

There is much to learn about this learn/enable/disable triad!

3.2.4.4 Menu / File / Options / Mouse

This item selects the mouse-interaction method.

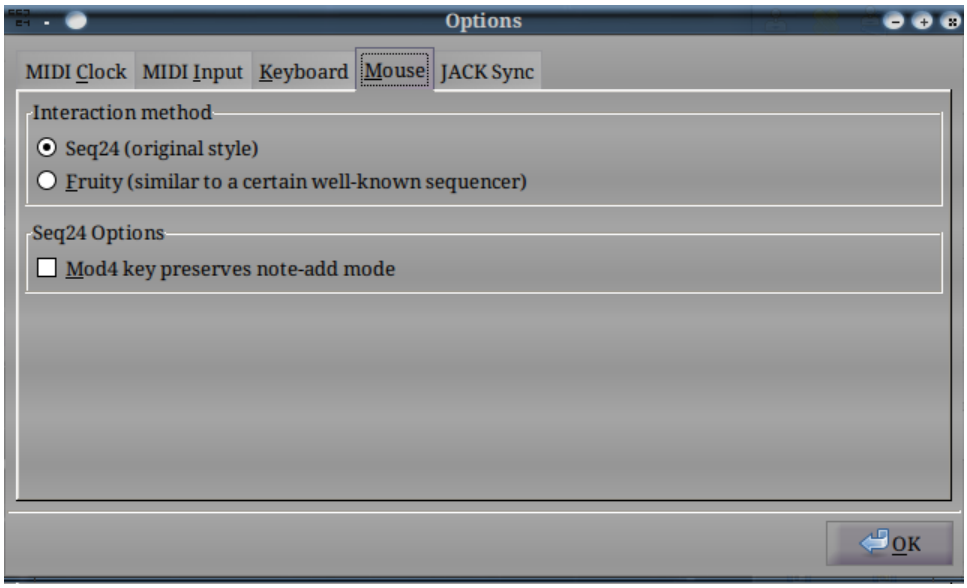

Figure 12: File / Options / Mouse (Condensed View)

The default method is **seq24 (original style)**. The alternate 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?

New: The Mod4 Right-Click Mode. In order to work better with certain trackpads, the "Seq24" mode of mouse interaction can be modified (only in the Pattern Editor at present) 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.

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).

TODO: This option is currently limited to the pattern editor window, and also needs some more work to see what can be done with the Mod4 key. But note that some *Sequencer64* windows can use the ctrl-left-click as a middle click.

3.2.4.5 Menu / File / Options / Jack Sync

This tab sets up options for JACK synchronization.

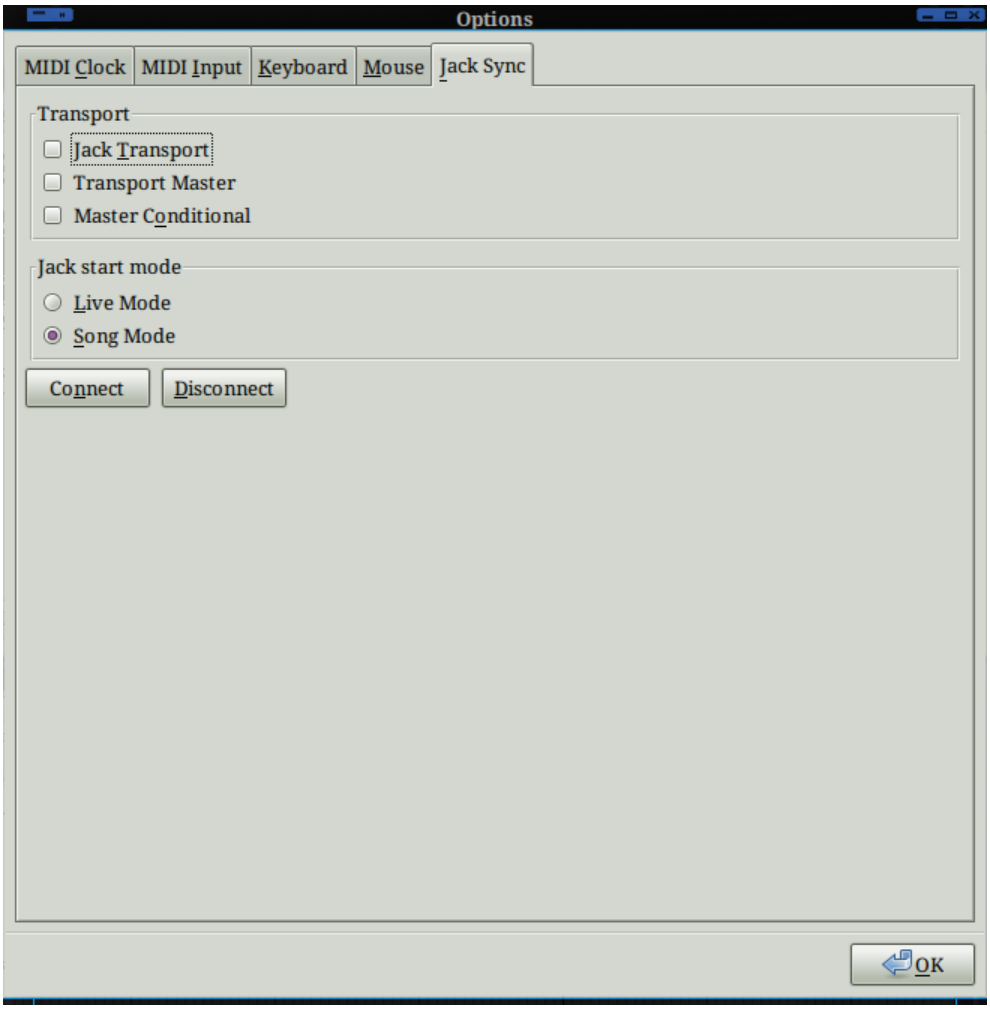

Figure 13: File / Options / Jack Sync

1. **Transport**
2. **Jack start mode**
3. **Connect**
4. **Disconnect**

1. Transport. This items collects the following settings:

- **Jack Transport**. Enables synchronization with JACK Transport.
- **Transport Master**. *Sequencer64* will attempt to serve as the JACK Master.
- **Master Conditional**. *Sequencer64* will fail to serve as the JACK Master if there is already a Master set.

2. Transport. This items collects the following settings:

- **Live Mode**. Playback will be in live mode. Use this option to allow muting and unmuting of patterns.
- **Song Mode**. Playback will use only the Song Editor's data.

3. **Connect.** Connect to JACK Sync.

4. **Disconnect.** Disconnect from JACK Sync.

3.3 Menu / View

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 42 ("Song Editor Window") on page 45

3.4 Menu / Help About...

This menu entry shows the "About" dialog.

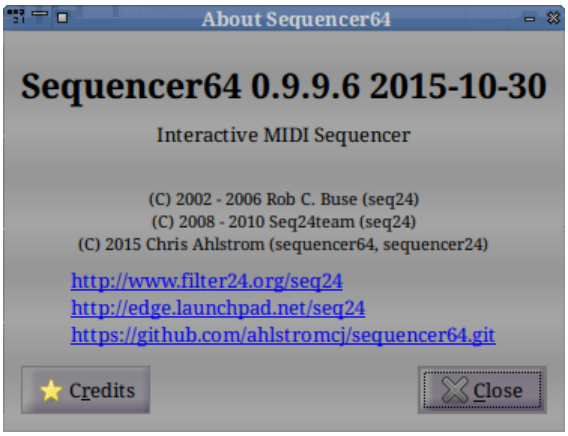

Figure 14: Help About

That dialog provides access to the credits for the program, including the authors and the project documentors.

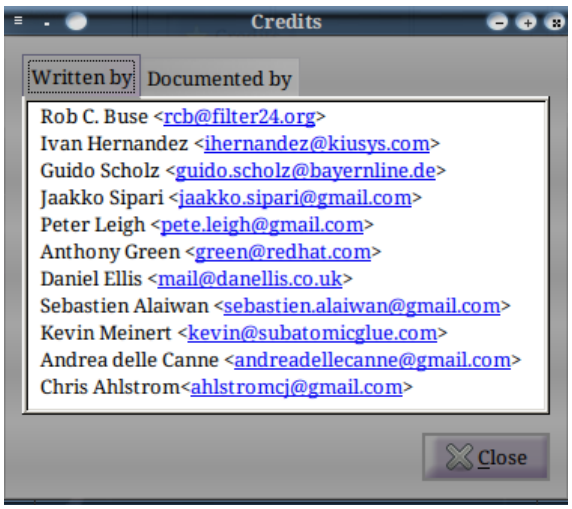

Figure 15: Help Credits

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

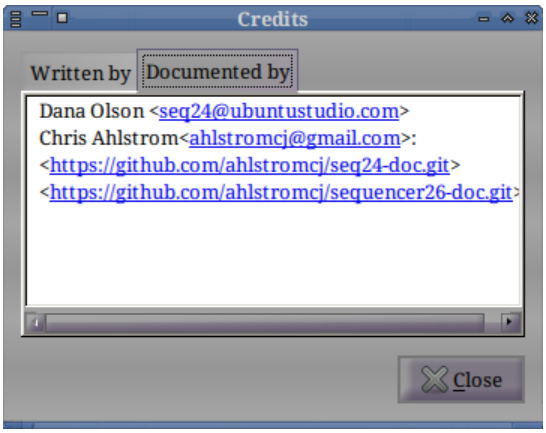

Figure 16: Help Documentation

Shows who has documented this project. Say, maybe "we" can get "our" name there someday! :-)

4 Patterns Panel

Sequencer64 works with the idea of patterns (loops) 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") on page 7. It is also called the "main window" or the "patterns window". It is here one manages a set of patterns (see section 2.1.13 ("Concepts / Terms / screen set") on page 10), manages the configuration, and opens the pattern or song editors.

When the Patterns Panel has the application focus, it puts *Sequencer64* in "live mode". The musician can control the playback and muting/unmuting of the song, while it is playing, from within this window.

For exposition, we break the Patterns Panel into a top panel, a pattern panel, and a bottom panel. Note that the *Sequencer64* main menu is discussed in section 3 ("Menu") on page 11.

4.1 Patterns / Top Panel

The top panel of the Pattern window is simple, consisting of the name of the program and a couple of controls.

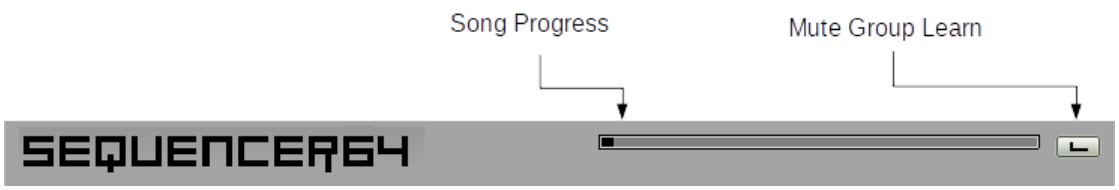

Figure 17: Patterns Panel, Top Panel Items

1. **Song Progress**
2. **Mute Group Learn**

1. 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. 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 have their own moving progress cursor, a tall thin line in each box. Unfortunately, this bar moves along even in patterns that have no events, only meta items such as the track name.

2. 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 of the patterns in that key.

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.

Group-learn is a modifier key to be pressed *together* with the toggle key, and the group on/off keys are there to enable/disable the whole feature, *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. When you ask the program to 'learn' the key, one can't use the Shift key, so (on Windows at least) one cannot use the "!" or other symbol keys. Similarly, make sure Caps Lock is off before starting the 'learn' process (as it won't recognise "q", only "Q").

Once that works, one can configure the MIDI settings in similar ways by assigning MIDI commands to toggle loops, using the 'on' option in the "rc" file. See section 7.1 ("Sequencer64 / MIDI Control Section") on page 50.

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

This toggling is done either by one of the *group toggle* keys or by a MIDI controller, both assigned in the `/.config/sequencer64/sequencer64rc` or `/.seq24rc` files.

A mute/unmute pattern (group) is stored by holding a *group learn* key (`Insert` by default) while pressing the corresponding *group toggle* key. There are also keys assigned to turn on/off the group functionality.

Remember that groups work with the playing screen set only. One must change the screenset and give it the command to make it the playing one (some set the Home key for this purpose). Everything is configurable "rc" file.

4.2 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 more 32 loops can be supported by *Sequencer64*. This group of 32 loops is called a "screen set", as discussed in section 2.1.13 ("Concepts / Terms / screen set") on page 10. 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 playing at a time, according to other notes we have found.

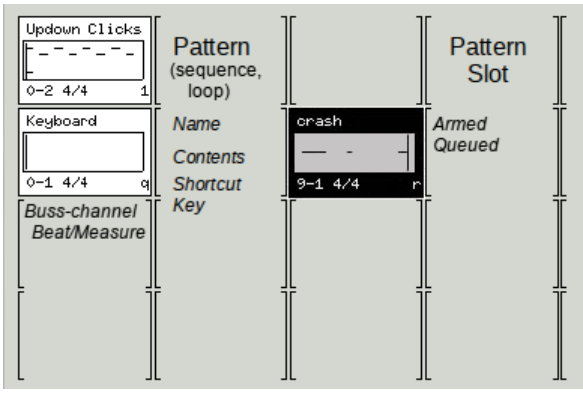

Figure 18: Patterns Panel, Main Panel Items

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

4.2.1 Pattern Slot

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

Figure 19: Empty Pattern, Right-Click Menu

1. **New**
2. **Paste**
3. **Song / Mute All 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.

2. Paste. Pastes a loop or pattern that was previously copied.

3. Song / Mute All Tracks. This item is the one item in the **Song** context menu; it mutes all tracks (or loops/patterns).

(We are not clear on exactly what it does. There is no change in visible status of any of the patterns in the patterns-panel.)

4.2.2 Pattern

A filled pattern slot is referred to informally as a pattern. A pattern is shown in the Pattern windows as a filled box with the following items of information in it:

- **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. **Bug:** However, an imported empty pattern will still (needlessly) scroll; perhaps such patterns can be marked as inactive.
- **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 3.2.4.3 ("Menu / File / Options / Keyboard") on page 18), 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.

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 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, if the Song Editor is the active window and was used to start the playback.

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:

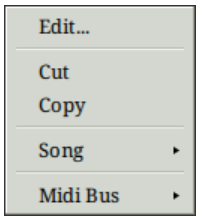

Figure 20: Existing Pattern, Right-Click Menu

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. **Cut**
3. **Copy**
4. **Song/**
5. **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. See figure 26 ("Pattern Edit Window") on page 32.

2. Cut. Deletes and copies an existing loop or pattern.

Bug: This operation seems to have no effect. The loop or pattern remains in place.

3. Copy. Copies an existing loop or pattern. The pattern can then be pasted elsewhere in the Patterns panel. See section 4.2.1 ("Pattern Slot") on page 26.

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

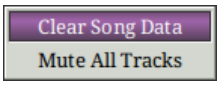

Figure 21: Existing Pattern, Right-Click Menu, Song

1. **Clear Song Data**
2. **Mute All Tracks**

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

2. Mute All Tracks. Selecting this filled-box right-click menu item causes... TODO. Cannot yet see that this does anything, NEEDS EXPERIMENTATION.

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

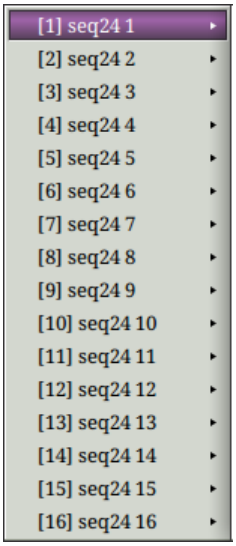

Figure 22: Existing Pattern, Right-Click Menu, MIDI Output Busses

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

Figure 23: Existing Pattern, Right-Click Menu, MIDI Bus Ports

4.2.3 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.

4.2.3.1 Pattern Keys

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

For each pattern, hitting its assigned keyboard 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.

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 pressed, on can then change the state of the

patterns to change how the song plays back. 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 generally taken over by the window manager, so there could be a need to change these items to some other keys.

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 following figure.

Queue also works for mute/unmute patterns (groups). In this case every sequence will toggle its status after its individual loop end.

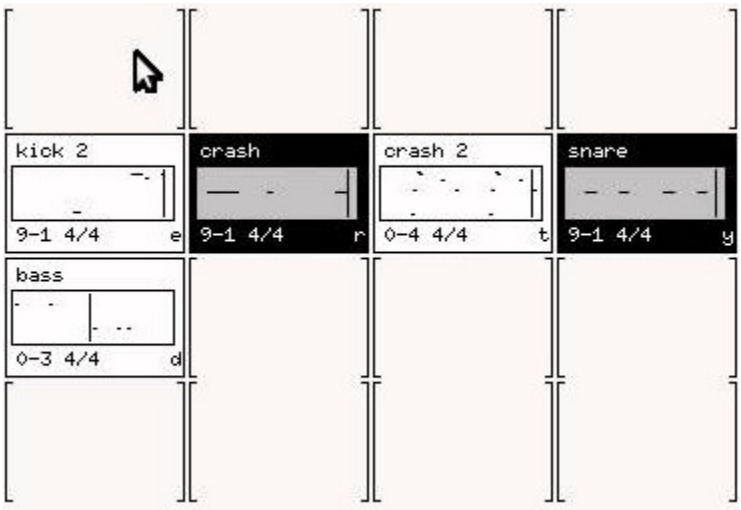

Figure 24: Pattern Coloration when Queued

Queue also works for mute/unmute patterns (groups); in this case every sequence will toggle its status after its individual loop end.

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 replace the right Ctrl key, unless one has (like the author) configured the window manager to use the Super key modifier to manipulate windows and applications *(laughter ensues)*.

Note that there is also a "replace" key, which is the left Ctrl key by default. 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.

Pressing the "keep queue" key (by default, the backslash key) assigned in the "rc" file. activates permanent queue mode until you use the temporary queue function again pressing `Right Ctrl`.

This key can be changed in the **File / Options / Keyboard / Keep queue** field.

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, tutorials about using arming, queueing, and snapshots, see references [14] and [15].

4.2.3.2 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.

Holding down `Left Ctrl` while selecting a pattern with a left click will mute all other patterns and turn on the selected pattern.

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.

4.3 Patterns / Bottom Panel

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

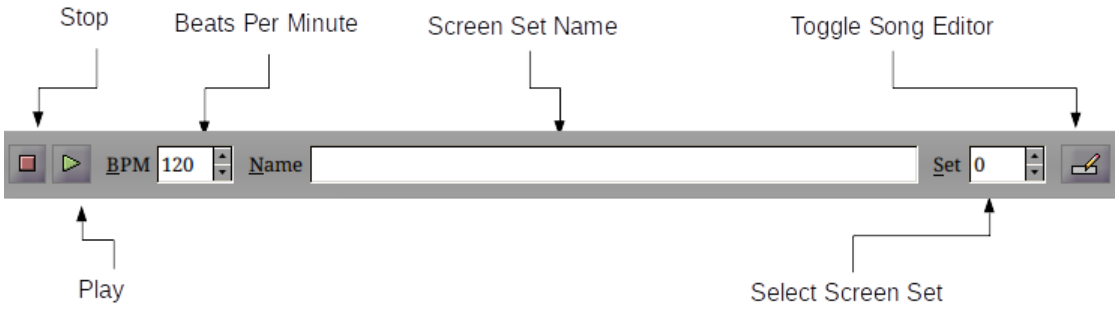

Figure 25: Patterns Panel, Bottom Panel Items

1. **Stop**
2. **Play**
3. **bpm**
4. **Name**
5. **Set**
6. **Toggle Song Editor**

1. Stop. The red squarebutton 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.

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

3. bpm. The spin widget adjusts the "Beats Per Minute" or BPM value. The range of this field is from 20 bpm to 500 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.

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

Bug: While one is typing in the name of the set in this field, the keystrokes will affect the panel window, causing playback to start and pattern boxes to be toggled!

5. 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.

Bug: While one is typing in the number of the set in this field, the keystrokes will affect the panel window as well.

6. Toggle Song Editor. Pressing this button toggles the presence on-screen of the Song Editor.

5 Pattern Editor

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

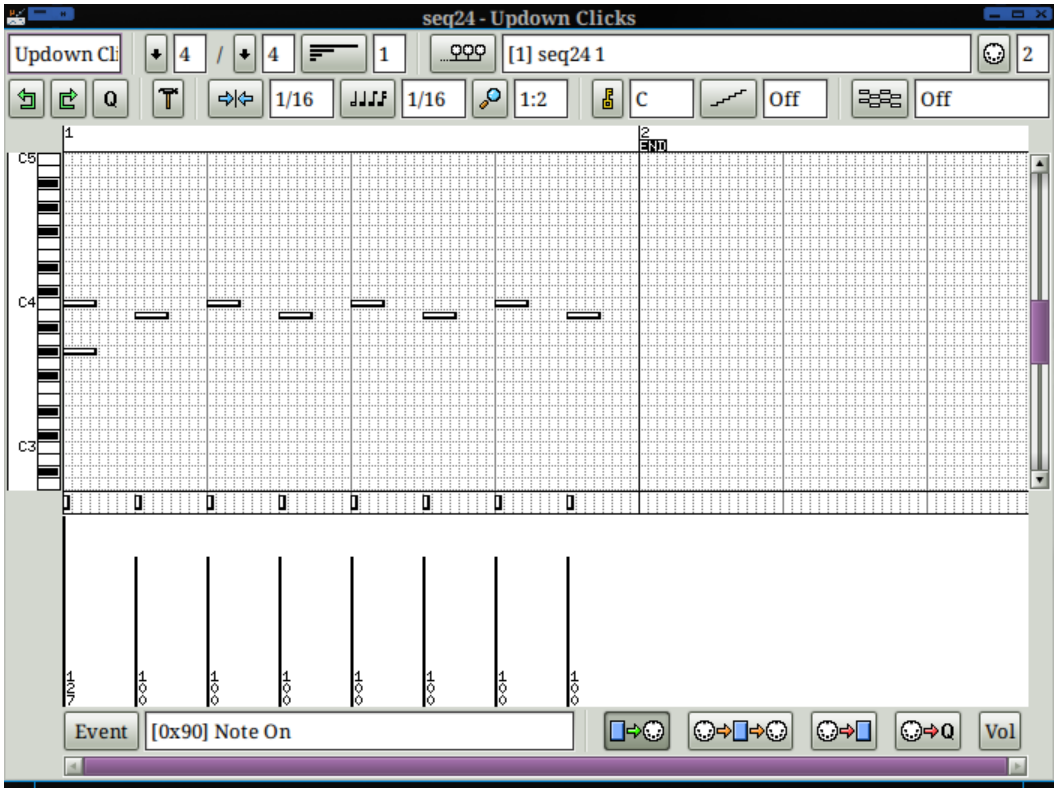

Figure 26: Pattern Edit Window

This dialog is quite complex. For exposition, we break it into 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**

5.1 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.

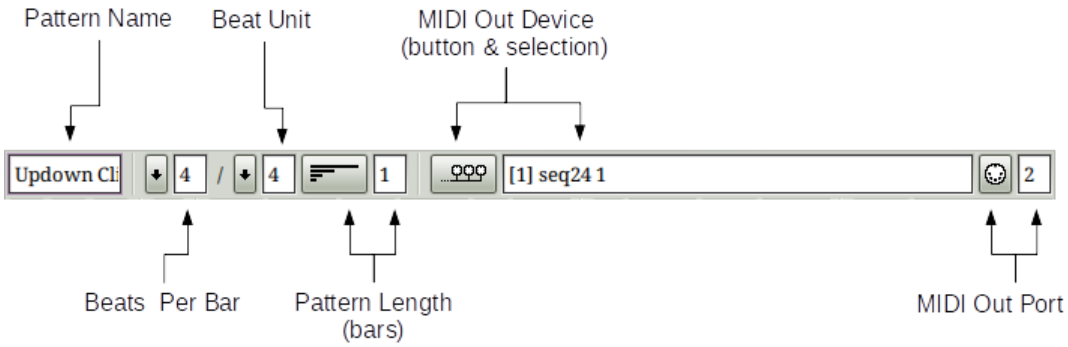

Figure 27: Pattern Editor, First Panel Items

1. **Pattern Name**
2. **Beats Per Bar**
3. **Beat Unit**
4. **Pattern Length**
5. **MIDI Out Device**
6. **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.

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. 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.

4. Pattern Length. Sets the length of the current pattern, in measures. The possible values range from 1 to 16, then 32, and 64.

(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.)

5. MIDI Out Device. This setting specifies one of the 16 MIDI output busses provided by *Sequencer64*. The settings look a lot like figure 22 ("Existing Pattern, Right-Click Menu, MIDI Output Busses") on page 28.

6. MIDI Out Port. This settings select the MIDI output channel, or port. The possible values range from 1 to 16.

5.2 Pattern Editor / Second Panel

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

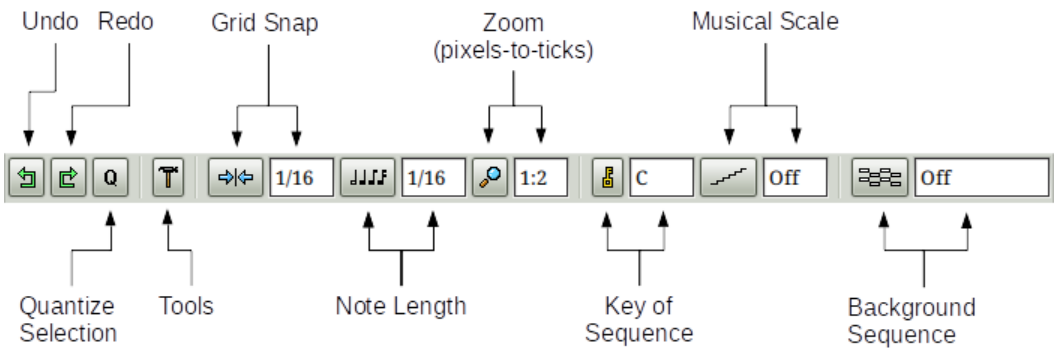

Figure 28: Pattern Editor, Second Panel Items

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**

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 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 doesn't seem to be a "Redo" key.

3. Quantize Selection. Pressing this button will quantize the selected events, presumably as per the **Grid Snap** setting.

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

Figure 29: 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:

- **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 a event *adds* that event to the selection.
- **Left Click Drag**. Pressing the left mouse button and dragging also lets one select multiple events and notes.
- **Ctrl Left Click Drag**. Pressing the `Ctrl` while left-click+dragging lets one make additional selections of multiple events and notes.

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

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

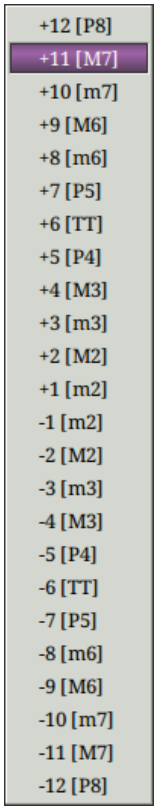

Figure 30: Tools, Transpose Selected Values

• 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.

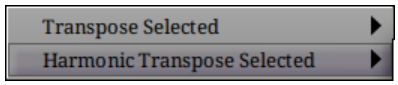

Figure 31: Tools, Two "Transpose" Menus

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:

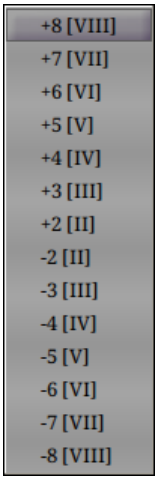

Figure 32: Tools, Harmonic Transpose Selected Values

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 what size they will be. 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. For example, 1:4 = 4 ticks per pixel. Supported values are 1:1, 1:2, 1:4, 1:8, 1:16, and 1:32.

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.

9. Musical Scale. Selects the desired scale for the pattern. Originally, only the following scales were supported: Off, Major, and Minor.

New: With *Sequencer64 v. 0.9.3*, the following scales are supported:

- **Off (chromatic)**
- **Major**
- **Minor**
- **Harmonic Minor**
- **Melodic Minor**

- **Whole Tone**

The new scales added in *Sequencer64* are **Harmonic Minor**, **Melodic Minor**, and **Whole Tone**.

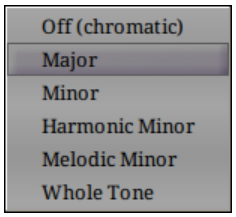

Figure 33: Available Scales

One can select which **Musical Scale** and **Key** the piece is in, and *Sequencer64* will grey out those keys on the piano-roll that are *not* in the selected scale for the selected key. This effect is shown for the C Major scale in the following figure:

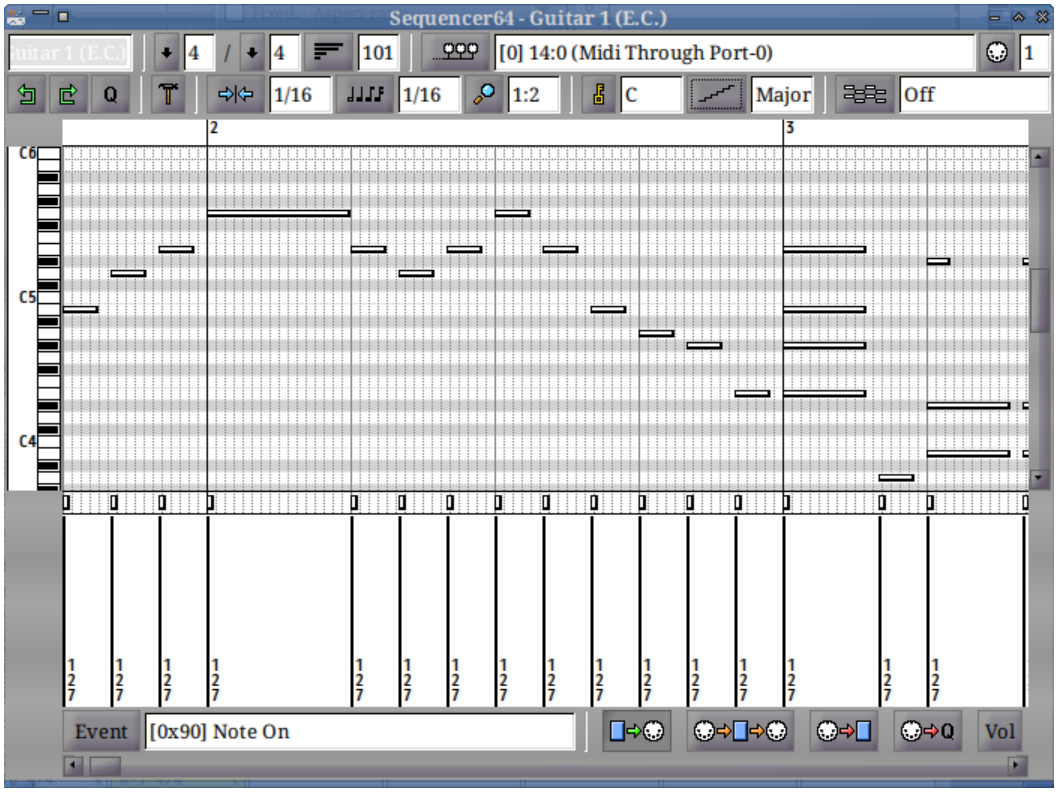

Figure 34: C Major Scale Masking

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.

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**]. Presumably, the 0 is a set number. Under the **0** entry, a menu like the following appears:

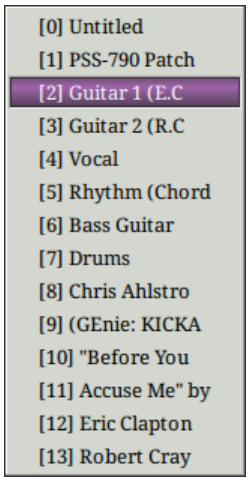

Figure 35: Sample Background Sequence Values

Once the desired pattern is selected from that list, it appears as gray notes, along with the notes that are part of the pattern. (Also note the orange selected notes and events in the following figure.)

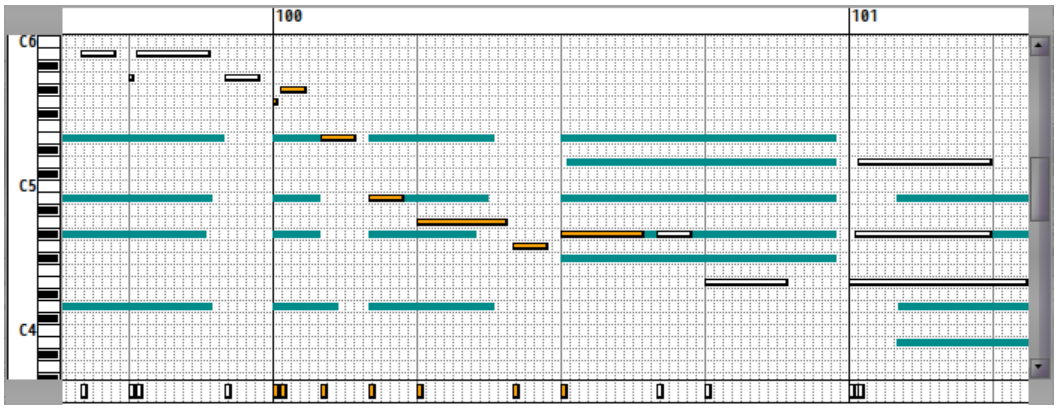

Figure 36: Background Sequence Notes

The gray notes shown represent a rhythm pattern.

5.3 Pattern Editor / Piano Roll

The piano roll is the heart of the pattern (loop, sequence) editor. While it 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. We tend to like the Logitech Marble Mouse, an ambidextrous USB trackball. It has four bottons, 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.

New: In addition, 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 3.2.4.4. 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.

5.3.1 Pattern Editor / Piano Roll Items

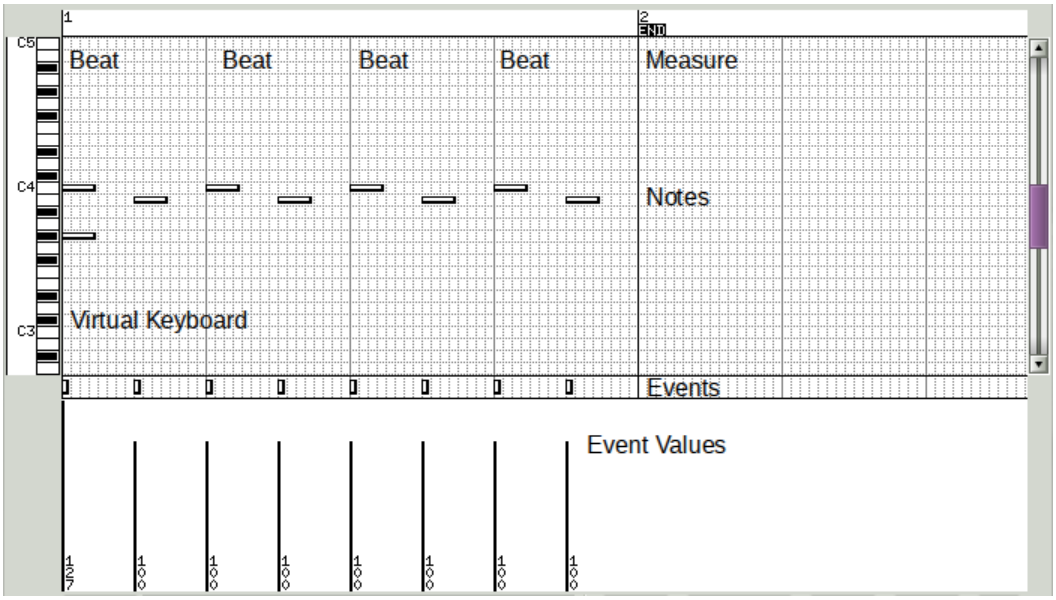

Figure 37: Pattern Editor, Piano Roll Items

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.

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 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 one will be drawing. It can be played with a mouse, to preview a short motif. It can show marks to indicate off-scale notes, to make them easy to avoid.

4. Notes. Musical notes are indicated by thick horizontal bars. Each bar provides a visual representation of the pitch of a note and the length of a note.

5. Events. 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 trying to edit or select events in that pane, in general, 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.)

6. Event Values. 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-clicking and dragging the mouse past each line, to chop it off at the given value. Easier to try it than explain it..

5.3.2 Pattern Editor / Event Editing

Note editing is a bit different with *Sequencer64*, since it requires two mouse buttons in many cases. There are some new laptop touchpads from FocalTech that have only one mouse button, and use positioning to determine if the click is a left-click or a right-click. The Linux drivers for this touchpad aren't sophisticated enough (as far as we know) to handle converting a two-fingered press properly. We've found that a good solution is to use a four-button USB trackball configued with an easy middle-button setup. It's easier than a touchpad, anyway. See the comments at the beginning of this section, too.

Note that, if only a middle-button is needed, ctrl-left-button will simulate that button. Also note the "Mod4 mode" for the right-click action, discussed elsewhere.

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

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

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" mode.

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.

To increase the duration of the note(s), keep dragging the mouse (with both buttons held) rightward. Note that, if one keeps dragging the mouse past the duration of a beat unit (e.g. a quarter note), then *multiple* notes are drawn, each one being the duration of a beat unit. This *auto-notes* facility can be convenient.

However, if one wants a *single long note*, first draw the short note, and then use one of the operations described in the next paragraph to change the length of the note. These operations also cause the note to be selected.

Pressing the **middle** mouse button ***or*** pressing the **ctrl left** mouse button in tandem will let one change the length of the note. If more than one note is selected, then the length of all selected notes is changed.

Warning: If one reduces the length of a note by more than its current length, the note will end up "infinitely" long.

The **left** mouse button lets one select multiple events which can then be clicked and moved, cut (`Ctrl-X`), copied (`Ctrl-C`), or pasted (`Ctrl-V`). When the notes are selected, one can delete them with the `Delete` key.

Pressing the `Ctrl` while left-click+dragging lets one make additional selections of multiple events and notes.

For the appearance of selected events (orange), see figure 38 ("Piano Roll, Selected Notes and Events") on page 41.

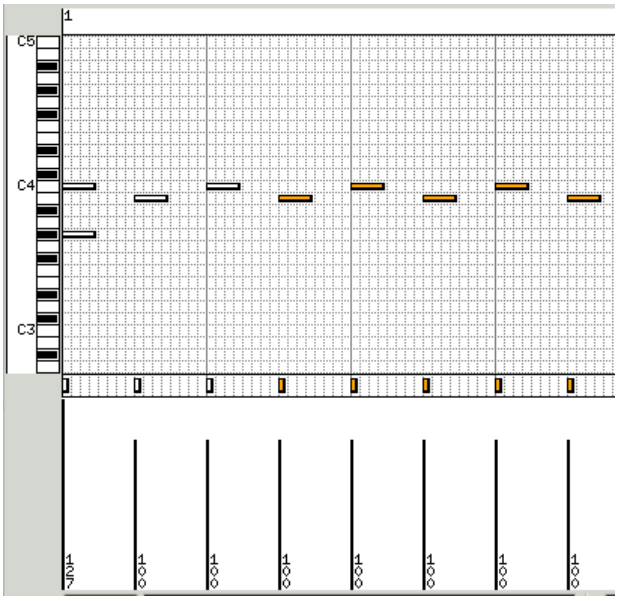

Figure 38: Piano Roll, Selected Notes and Events

Pressing the left button on a note or a event deselects all other notes or events, and selects the item clicked on.

Pressing the `Ctrl` key and the left button on a note or a event *adds* that event to the selection.

Note that pressing the left mouse button and dragging also lets one select multiple events and notes. Pressing the `Ctrl` while left-click+dragging lets one make additional selections of multiple events and notes.

Note that pressing the left mouse button and dragging also lets one The **Tools** button described in section 5.2 ("Pattern Editor / Second Panel") on page 33can also be used to modify selections.

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

Once a selection of notes is made, one can use the Shift-middle-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 the length of a note by more than its current length, the note will end up "infinitely" long.

5.3.2.2 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. *Note: it is currently not possible to move them to positions smaller than the beat size. Is there a way to do it?*

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.

5.4 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*.

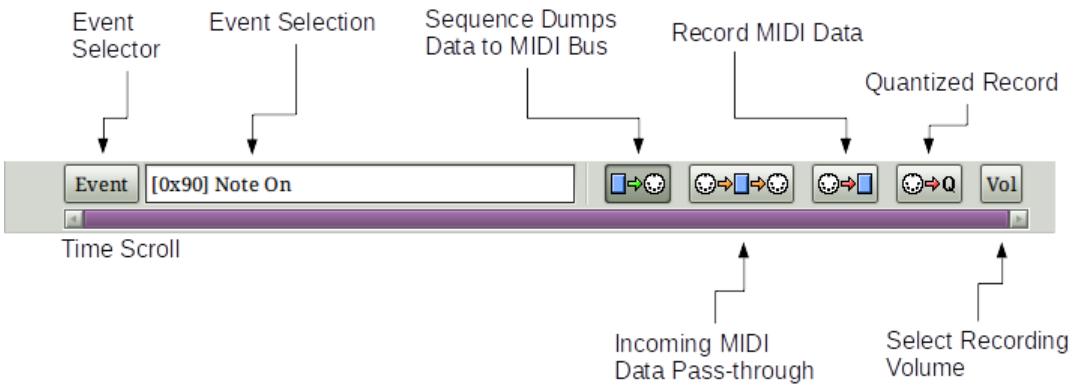

Figure 39: Pattern Editor, Bottom Panel Items

1. **Event Selector**
2. **Event Selection**
3. **Time Scroll**
4. **Data To MIDI Buss**
5. **MIDI Data Pass-Through**
6. **Record MIDI Data**

7. **Quantized Record**
8. **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.

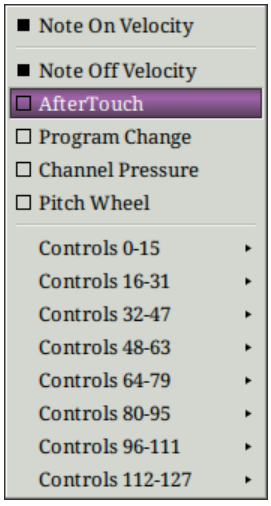

Figure 40: Pattern Editor, Event Button Context Menu

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 `$HOME/.config/sequencer64/sequencer64rc` (legacy mode: `$HOME/.seq24usr`), to make it match one's instrument. See section 8 ("Sequencer64 User Configuration File") on page 59.

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

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

4. 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.

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

6. 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.

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

8. Vol. This button allows controlling the volume of the recording.

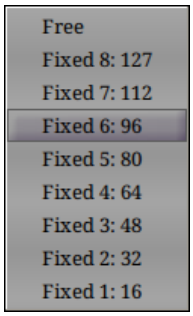

Figure 41: Pattern Recording Volume Menu

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.

6 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*. It also provides the "song mode" of *Sequencert24*, as opposed to the "live mode" provided by the Patterns Panel.

When the Song Editor has the focus of the application, it takes over control from the Patterns Panel. The Song Editor controls now control 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.

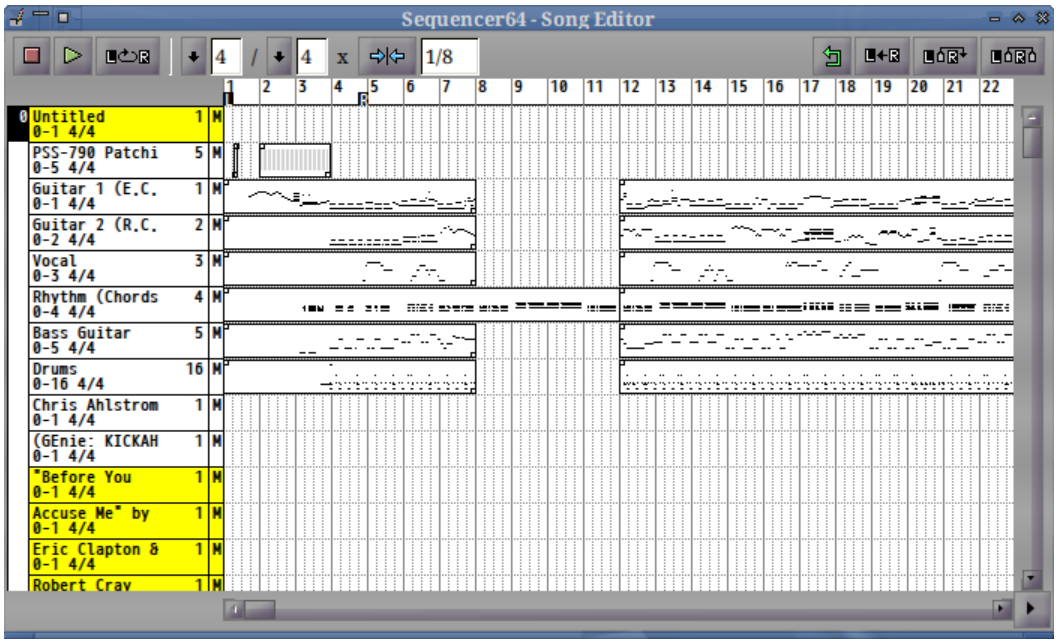

Figure 42: Song Editor Window

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 and the rest of the window.

6.1 Song Editor / Top Panel

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

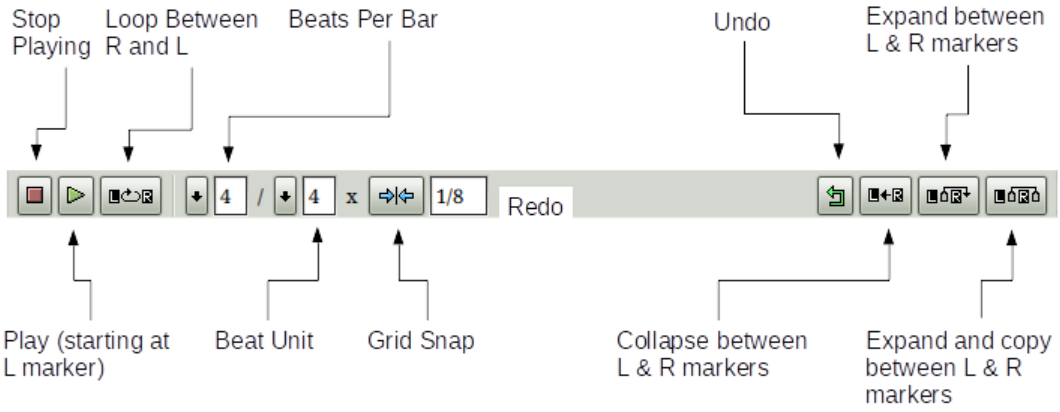

Figure 43: Song Editor / Top Panel Items

1. **Stop**
2. **Play**

3. **Play Loop**
4. **Beats Per Bar**
5. **Beat Unit**
6. **Grid Snap**
7. **Undo**
8. **Collapse**
9. **Expand**
10. **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.

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

7. 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. There is no Redo button in the Song Editor.

8. 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.

9. 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.

10. 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.

6.2 Song Editor / Arrangement Panel

The arrangement panel is the middle section shown in figure 42 ("Song Editor Window") on page 45. It is also known as the "piano roll" of the song editor. Here, we zero in on its many features.

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 might need to 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.

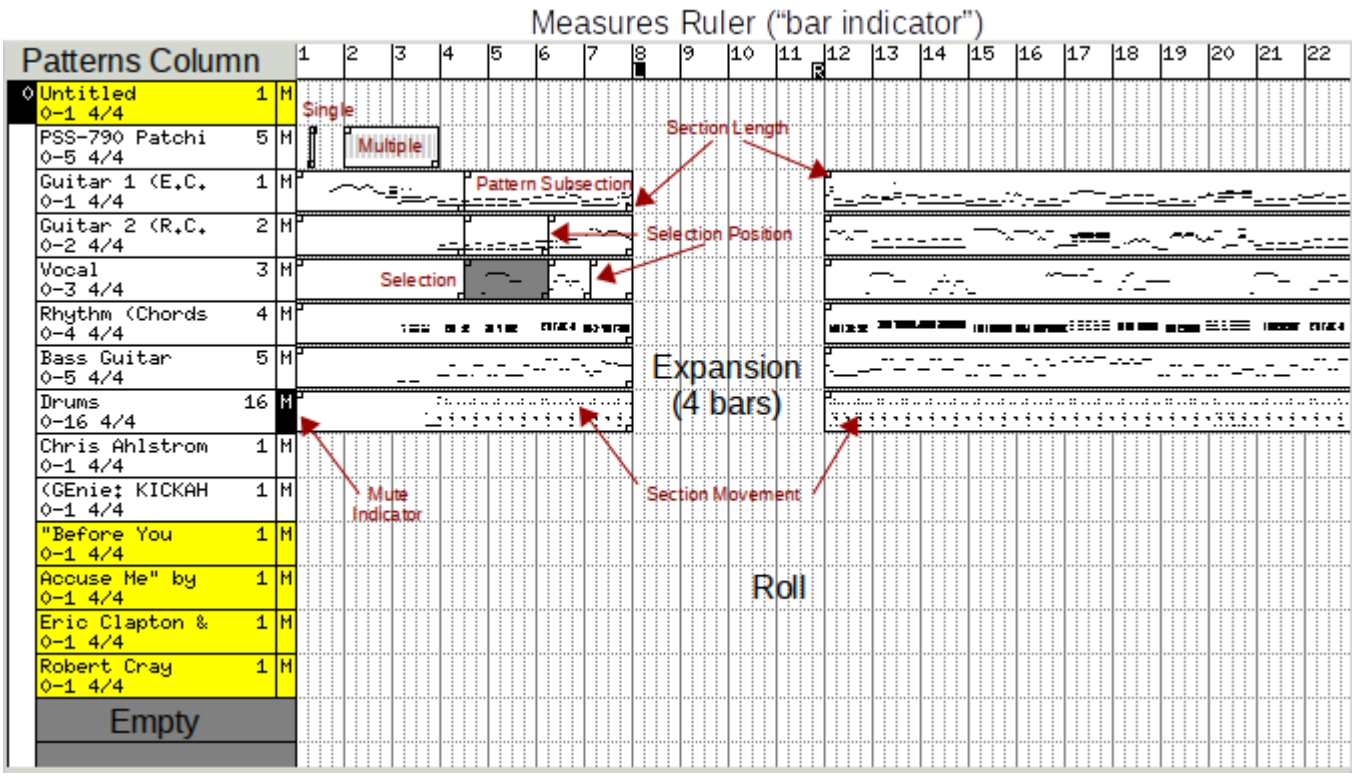

Figure 44: Song Editor Arrangement Panel, Annotated

It 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.

6.2.1 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.
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.

Right-clicking on the pattern name or the "M" button brings up the same pattern editing menu as discussed in section 4.2.2 ("Pattern") on page 26. Recall that this context menu has the following entries: **Edit...**, **Cut**, **Copy**, **Song**, and **Midi Bus**.

6.2.2 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 44 ("Song Editor Arrangement Panel, Annotated") on page 47:

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 the corner 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.
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.

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 3.2.4.4. 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.

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.

6.2.3 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 drop an **L anchor** on the measures ruler. Right-click in the measures ruler to drop an **R anchor** on the measures ruler.

Once these anchors are in place, 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.

7 Sequencer64 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 `sequencer64rc` file in your home directory:

```
/home/ahlstrom/.config/sequencer64/sequencer64rc
```

It contains the the data for remote MIDI control, keyboard control, and MIDI clock.

Sequencer64 will overwrite the `sequencer64rc` file upon quitting. One should therefore quit *Sequencer64* before doing manual modifications to the `sequencer64rc` file.

7.1 Sequencer64 / MIDI Control Section

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 `sequencer64rc`, and begins with an "INI" group marker `[midi-control]`.

THe MIDI Control section is implicitly 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.
2. **mute in group**. Consists of 32 lines, one for each pattern box shown in the Pattern window.
3. **automation group**. Each item in this group consists of one line.
    1. **bpm up**. Consists of one line.
    2. **bpm down**. Consists of one line.
    3. **screen-set up**. Consists of one line.
    4. **screen-set down**. Consists of one line.
    5. **mod replace**. Consists of one line.
    6. **mod snapshot**. Consists of one line.
    7. **mod queue**. Consists of one line.
    8. **mod gmute**. Consists of one line.
    9. **mod glearn**. Consists of one line.
    10. **screen-set play**. Consists of one line.

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

```
[midi-control]
74

# pattern group
 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]
 2  [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]

# mute in group
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 first number is the pattern/sequence number in the main window, which ranges from 0 to 31. Each set of brackets corresponds to a MIDI filter. The MIDI filter in the leftmost brackets is the *toggle* filter. The MIDI filter in the middles brackets is the *on* filter. The MIDI filter in the rightmost brackets is 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 bracket is as follows:

[OPR INV STAT D1 D2min D2max]

where

- **OPR = on/off**
- **INV = inverse**
- **STAT = MIDI status byte** (channel ignored)

- **D1 = data1**
- **D2min = data2 min**
- **D2max = data2 max**

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

The **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 last three fields describe the range of data that will match.

7.1.1 Sequencer64 / 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 **toggle** field is off (inactive).

The **on** field is on (active). Inverse is inactive. The **MIDI status byte**, 144, is 0x90 (hex), which is a Note On event on channel 0. However, the channel is ignored. **data1** is 0, and **(**data2) can range from 0 to 127.

The **off** field is on (active). The **MIDI status byte**, 128, is 0x80 (hex), which is a Note Off event on channel 0. Again, the channel is ignored. **data1** is 0, and **data2** can range from 0 to 127.

So, basically, pattern 1 starts when any Note On is received, and it stops when any Note Off is received.

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.

7.1.2 Sequencer64 / 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 (see after).

So, this part of the MIDI Control section is used for muting and unmuting (and toggling) a group of patterns.

7.1.3 Sequencer64 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. TODO. Explain the concept of replacement.

6. mod snapshot. This item provides a way to automate snapshots. TODO. Explain the concept of snapshots.

7. mod queue. This item provides a way to automate queueing. TODO. Explain the concept of queue.

8. mod gmute. This item provides a way to automate group-muting. Explain the concept of snapshots.

9. mod glearn. This item provides a way to automate group-learning. TODO. Explain the concept of group-learning.

10. screen-set play. This item provides a way to automate screen set play. TODO. Explain the concept of screen set play.

7.2 Sequencer64 / 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
 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 `sequencer64rc`.

7.3 Sequencer64 / 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 3.2.4.1 ("Menu / File / Options / MIDI Clock") on page 15

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
```

7.4 Sequencer64 / 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
# Key #, Sequence #
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
```

7.5 Sequencer64 / Keyboard Group Section

This section is the same as [**keyboard-control**], but to control groups. 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:

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 3.2.4.3 ("Menu / File / Options / Keyboard") on page 18.

```
[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
```

7.6 Sequencer64 / JACK Transport

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 3.2.4.5 ("Menu / File / Options / Jack Sync") on page 21.

```
[jack-transport]

# --jack_transport: Enable sync with JACK Transport.  Sequencer64 will sync
# to JACK transport if the JACK server is available.
0

# --jack_master: Sequencer64 will attempt to serve as JACK Master.
0

# --jack_master_cond: Sequencer64 will fail to be JACK master if there is
# already a JACK master set.
0

# --jack_start_mode n
# 0 = Playback will be in live mode.  Use this value to allow muting
#     and unmuting of loops.
# 1 = Playback will be in performance mode.  Playback will use the song
#     editor's data.  When Sequencer64 is synced to JACK, the playback command
#     comes from the JACK server.  Sequencer64 is in performance mode by default.
1
```

7.7 Sequencer64 / Other Sections

This configuration item is the same as the **Clock Start Modulo** option described in paragraph 3.2.4.1 ("Menu / File / Options / MIDI Clock") on page 15.

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

This configuration item is the same as the **MIDI Input** tab described in paragraph 3.2.4.2 ("Menu / File / Options / MIDI Input") on page 17. 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
0 0             # [0] seq24 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
```

Turning on the manual-alsa-ports option is necessary if one wants to use *Sequencer64* with JACK.

This configuration item is the same as the **Mouse** tab described in paragraph 3.2.4.4 ("Menu / File / Options / Mouse") on page 20.

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

[interaction-method]
0

# 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.

0
```

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.

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

```
[last-used-dir]

# Last used directory.
```

8 Sequencer64 User Configuration File

The *Sequencer64* configuration file was called `.seq24usr`, 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, it will generate a `sequencer64usr` file in your home directory:

```
/home/ahlstrom/.config/sequencer64/sequencer64usr
```

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 `sequencer64rc` 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.

8.1 Sequencer64 User / 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 buses
```

This means that the `sequencer64usr` 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
```

8.2 Sequencer64 User / 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 "rc" 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.

Sometime we would like to create a section that sets up the *Yoshimi* 1.3.5+ software synthesizer as an instrument.

9 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.

Sequencer64 is a real-time MIDI sequencer. It was created to provide a very simple interface for editing and playing MIDI 'loops'.

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

Sequencer64 accepts the following options, plus an optional name of a MIDI file.

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

`-v --version`
Display the program version.

`-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 `/.seq24rc` and `/.seq24usr` files, instead of the new `/.config/sequencer64/sequencer64rc` and `/.config/sequencer64/sequencer64usr` 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 binary to make that happen).

`-L --lash`
New: If LASH support is compiled into the program, this option enables it.

`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 ALSA ports.

`-s --showmidi`
Dumps incoming MIDI to the screen.

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

`N/A --pass_sysex`
Passes any incoming SYSEX messages to all outputs.

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

`-k --show_keys`
Prints pressed key value.

`-x --interaction_method [number]`
Select the mouse interaction method. 0 = seq24 (the default); and 1 = fruity loops method.

`-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.

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

`$HOME/.seq24rc` holds the user settings for *Sequencer64*.

The old 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 was written by Chris Ahlstrom ¡ahlstromcj@gmail.com¿. *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 0.9.3` `September 1 2015` `Sequencer64(1)`

10 Proprietary Track Format and Other MIDI Notes

10.1 Legacy Proprietary Track Format

Before we get to the last, proprietary track, note that the tracks that precede it include the SeqSpec ("sequencer-specific", sort of) control tags shown in table 1 ("SeqSpec Items in Normal Tracks") on

Table 1: 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_timesig	24 24 00 06 00 00 00 00
c_triggers_new	24 24 00 08 00 00 00 00

page 64. These control tags are global constants in the *Seq24* source code, ranging from 0x24240001 to 0x24240010..

Note that these tags (not created by the application, and not present in the proprietary track, but 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 5 ("Application Support for MIDI Files") on page 68, most applications accept the shortcut legacy format, but *midicvt* does not.

So, as a "bug" fix, we want to be able to write and read this information properly in *Sequencer64*. We also need to be able to read legacy Seq24 MIDI files.

Any way, 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 busscount were 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 2 ("SeqSpec Items in Legacy Proprietary Track") on page 65

Table 2: 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 3 ("SeqSpec Items in New Proprietary Track") on page 66.

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
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.

Table 3: 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)

10.2 MIDI Information

This section just provides some useful, basic information about MIDI data.

10.2.1 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
```

10.2.2 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.

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.

Table 4: 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

10.2.3 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 5 ("Application Support for MIDI Files") on page 68, 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?)

11 Summary

In summary, we can say that you will find *Sequencer64* intriguing.

There are some topics that this document does not yet treat ...:

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.

Table 5: 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

12 References

The *Yoshimi* seq24 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] JACK team *JACK Audio Connection Kit* http://jackaudio.org/ 2015.

[6] LinuxSynths team, briandc@linuxsynths.com *A Sonic Palette on the Linux Platform.* http://www.linuxsynths.com/ 2015.

[7] Dave Phillips *At the Sounding Edge: Introducing seq24.* http://www.linuxjournal.com/article/8304 Linux Journal, May 12, 2005.

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

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

[10] Seq24 Team. *The home site for the Sequencer64 looping sequencer.* http://www.filter24.org/seq24/download.html 2010.

[11] Chris Ahlstrom. *A continuation of the Sequencer64 project as "Sequencer64".* https://github.com/ahlstromcj/sequencer64/ 2015.

[12] Kevin at subatomicglue.com *Subatomic Mods for Seq24 Win32* http://www.subatomicglue.com/seq24/ 2010.

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

[14] Author's name. *Sequencer64 Tutorial Video, Part 1.* http://wootangent.net/2010/10/linux-music-tutorial-seq24-part-1/ 2010.

[15] Author's name. *Sequencer64 Tutorial Video, Part 2.* http://wootangent.net/2010/10/linux-music-tutorial-seq24-part-2/ 2010.

[16] Yoshimi team abrolag@users.sourceforge.net *The download site for the Yoshimi software synthesizer.* http://yoshimi.sourceforge.net/ 2015.

[17] Yoshimi team *The alternate location for the Yoshimi source-code.* https://github.com/abrolag/yoshimi/ 2015.

[18] Chris Ahlstrom *A Yoshimi User Manual.* https://github.com/ahlstromcj/yoshimi-doc/ 2015.

[19] Chris Ahlstrom *A Yoshimi Cookbook.* https://github.com/ahlstromcj/yoshimi-cookbook/ 2015.

[20] Mark McCurry, Paul Nasca (ZynAddSubFX team) *The download site for the ZynAddSubFX software synthesizer.* http://zynaddsubfx.sourceforge.net/ 2015.

Index