MIDIcvt/MIDIcvtpp User's Manual

0.3.3.7

Generated by Doxygen 1.8.11

Contents

1 midicvt and midicvtpp

Author(s) Chris Ahlstrom 2015-08-14

1.1 Introduction

MIDIcvt (`midicvt`) is a refactoring, augmentation, and documentation of the related *midi2text* and *midicomp* projects. The purpose of these projects was to convert MIDI to and from a human-readable, machine-parsed text format, for easy modifications to MIDI files using standard tools.

The text representation is chosen such that it is easily recognized and manipulated by programs like *sed*, *awk*, or *perl*. Yet it is also human-readable so that it can be manipulated with an ordinary text editor.

`midicvt` is a C program that does essentially the same task, with some minor upgrades.

`midicvtpp` is a C++ program that incorporates the functionality of `midicvt`, but adds the ability to do some canned MIDI-to-MIDI conversions using "INI" files to control the conversions. These canned conversions mean that the use can avoid learning scripting in order to perform certain conversions.

This document is based, in part, on documentation provided by the *midi2text* project at the https://code.google.com/p/midi2text/ site. Here is the information on the author of that program.

```
Piet van Oostrum, Dept of Computer Science, Utrecht University,
P.O. Box 80.089, 2508 TB Utrecht, The Netherlands
email: piet@cs.ruu.nl
```

Please check out his project and peruse it. We're not attempting to support Windows or Mac OSX at this time.

Those programs use the midifile library written by Tim Thompson (tjt@blink.att.com) and updated by Michael Czeiszperger (mike@pan.com). However, there were some bugs in the write code, and Piet and ourselves added some features that we needed. He also changed some of the names to cope with the 7-character limit for external identifiers in the Sozobon compiler. However, we changed the coding conventions to use more recent C and C++ standards. The new library is called `midifilex`.

Piet compiled the programs on an Atari ST with the Sozobon compiler and the dlibs library. The scanner is generated using flex 2.3. The output of flex (t2mflex.c) is included for those that do not have flex. The module yyread.c is a flex library module that you need on TOS (and on MSDOS). The corresponding makefile is makefile.st. For Unix use makefile.unx. For Borland C on MSDOS use makefile.bcc. For Microsoft C on MSDOS makefile.msc. The makefiles may need minor changes for other systems.

There don't seem to be any make-files at Piet's site. Luckily, it is easy to build his code from the command-line.

Licensing (GPL) for this project is defined in greater detail at the end of this document.

Related projects:

- https://github.com/markc/midicomp A version of the midicomp program. Dead sites for this same project include http://alsa.opensrc.org/MidiComp and http://midicomp.opensrc.org/.
- https://code.google.com/p/midi2text/ The midi2text project, providing the mf2t and t2mf programs, as described above.
- http://www.midiox.com/ Windows versions of the mf2t and t2mf programs.
- http://www.fourmilab.ch/webtools/midicsv/ Converts MIDI to CSV files. Not to be confused with our project's usage of CSV.

2 midicvt

Author(s) Chris Ahlstrom 2015-08-19

2.1 Introduction to midicvt

The C program `midicvt` lets one convert between text and MIDI formats, and leverage scripting to transform MIDI files in infinite ways.

2.2 Overview of midicvt

You can run the command

```
midicvt --help
```

in order to see the following brief explanation.

```
 -2  --m2m       Convert MIDI to MIDI (testing only in midicvt).
 -c  --compile   Flag to compile ASCII input into SMF/MIDI.
 -d  --debug     Send any debug output to stderr.
 -f  --fold [N]  Fold SysEx or SeqSpec data at N (default 80) columns.
 -i  --input [F] Specify input file (replaces stdin).  Default file-name is
                 'out.mid' or 'out.asc', depending on --compile option.
 -m  --merge     Collapse continued system-exclusives.

 -n  --note      Show note on/off value using note+octave.
 -o --output [F] Specify output file (replaces stdout). Default file-name
                 is 'out.asc' or 'out.mid', depending on --compile option.
 -t  --time      Use absolute time instead of ticks.
 -v  --verbose   Output in columns with --notes on.
 -r  --report    Write detailed information to stderr (debugging).
 --version       Show the version information for this program.
 --mfile         Write ASCII using 'MFile' instead of 'MThd' tag.
 --mthd          Write ASCII using the 'MThd' tag (default).  The program
                 can read either tag.
 --strict        Require that all tracks are marked with 'MTrk'.  By
                 default, tracks with other names can be processed.
 --ignore        Allow non-Mtrk chunks, but do not process them.
                 Per the MIDI specification, they should be ignored,
                 but midicvt otherwise treats them like tracks.

To translate a SMF file to plain ASCII format:

    midicvt midi.mid                    View as plain text.
    midicvt -i midi.mid [ -o ] midi.asc  Create a text version.
    midicvt midi.mid > midi.asc          Create a text version.

To translate a plain ASCII formatted file to SMF:

    midicvt -c midi.asc midi.mid         Create a MIDI version.
    midicvt -c midi.asc -o midi.mid      Create a MIDI version.

    midicvt midi.mid | somefilter | midicvt -c -o midi2.mid

 It is recommended to always use -i/--input and -o/--output to specify
 the input and output file-names.
```

The next section discusses each option in more detail.

2.3 Usage of midicvt

The basic use case is the conversion of a MIDI file to text, dumping the output to standard output. This output can be redirected to an output file.

```
$ midicvt midi.mid
   (lots of output)
$ midicvt midi.mid > midi.asc
```

We've made some extensions to the program, so that it is better, in general, to be specific about the input by using the -i or –input options, and to be specific about the outut by using the -o or –output options.

```
$ midicvt --input midi.mid --output midi.asc
```

To convert a file from the ASCII format back to MIDI, use the -c or –compile option:

```
midicvt --compile midi.asc midi.mid
midicvt --compile midi.asc --output midi.mid
```

You can also pipe the ASCII output to a filter program and then to a new MIDI file:

```
midicvt midi.mid | somefilter | midicvt --compile --output midi2.mid
```

However, we may have broken this facility. Don't be suprised if it doesn't work yet.

Also look at the `tests/test_script` to see some examples of using *midicvt*.

The next sections discuss each option in more detail.

2.3.1 midicvt --input

The -i or –input option specifies the input file without any chance for ambiguity. Unless you pipe the output through an ASCII filter, you should prefer using this option, as opposed to specifying the input file by its position as first file on the command-line.

2.3.2 midicvt --output

The -o or –output option specifies the output file without any chance for ambiguity. Unless you pipe the input from an ASCII filter, you should prefer using this option, as opposed to specifying the output file by its position as second file on the command-line.

2.3.3 midicvt --mfile

Midicvt has recently been modified to be able to output ASCII files that start with the "MThd" tag, rather than the "MFile" tag. If you want your ASCII files to be compatible with older programs, then use this option.

The current version of midicvt can read either variant of ASCII file.

2.3.4 midicvt --mthd

This option is the opposite of –mfile, and is now the default option.

2.3.5 midicvt --m2m

The -2 or –m2m option is a new feature of the C program. It doesn't do much but demonstrate that we can convert a file from MIDI to ASCII to MIDI. This feature is used more fully in the C++ program *midicvtpp*.

Note that this conversion will often apply some fixes to broken MIDI files, so that the output MIDI file is not the same as the input MIDI file. (However, both will generate the same ASCII output.)

2.3.6 midicvt --compile

The -c or –compile option converts an ASCII file generated by `midicvt -input` back into a MIDI file.

Note that, even if the ASCII file has not been edited, this process can actually fix minor issues in the MIDI file, and so the resultant MIDI file might not be identical to the original.

2.3.7 midicvt --debug

The -d or –debug option simply sends additional output to stderr, as an aid to troubleshooting. Also see the -r or –report option.

2.3.8 midicvt --fold

The -f or –fold option folds the ASCII output of generated SysEx data or SeqSpec (sequencer-specific) data for easier reading. Provide the number of columns at which to fold the output. If this number is not provided, the default value is 80 columns.

This option was useful when working on a fork (Sequencer24) of the Seq24 live-looping sequencer, and noting that the files saved by Seq24 caused errors in midicvt because they were not quite MIDI-compliant. Once we got Sequencer24 to write MIDI-compliant files, we saw that some of its sequencer-specific "proprietary" sections were very long. So this option proved useful.

See the following projects:

- http://www.filter24.org/seq24
- http://edge.launchpad.net/seq24
- https://github.com/ahlstromcj/sequencer24.git

That last URL is not yet in existence, but will be soon.

2.3.9 midicvt --merge

The -m or –merge option collapses continued system-exclusive message into one system-exclusive message (we think).

2.3.10 midicvt --note

The -n or –note option displays note values in note notation (e.g. a4#) as opposed to just a MIDI note number.

2.3.11 midicvt --strict

The midicvt application now, by default, tries to process non-MTrk chunks as if they were MTrk chunks. This allows the MIDI-compliant formats of certain sequencers to be processed by midicvt when converting to ASCII. (However, conversions in the opposite direction might now work.)

The –strict option causes the old behavior, where an error is reported and processing may abort.

2.3.12 midicvt --ignore

The –ignore option is intermediate between the now-default permissive behavior of midicvt and the –strict behavior that restores the old functionality.

This option allows non-MTrk chunks to be handled, but no output is generated for those non-MTrk chunks.

2.3.13 midicvt --time

The -t or –time option displays time in an expanded notation.

2.3.14 midicvt --verbose

The -v or –verbose option displays more information, and uses longer names for the various MIDI events.

2.3.15 midicvt --report

The -r or –report option is a new feature used mostly by the developer. It dumps the MIDI information to standard-error in a very verbose format. You won't have much use for this option.

2.3.16 midicvt --version and --help

These options tell you something about the program.

3 midicvtpp

Author(s) Chris Ahlstrom 2016-04-17

3.1 MIDI to MIDI Mode

The C program `midicvt` lets one convert between text and MIDI formats, and leverage scripting to transform MIDI files in infinite ways.

However, some people might not care to work up a good script. Therefore, the C++ program `midicvtpp` extends `midicvt` to add a MIDI-to-MIDI mode that lets the transformations to be made be specified in an "INI" file.

An INI file is a file with a format familiar to some from the days of DOS. Examples of such files can be found in the `doc` directory of this project. They are easier to work with than XML files, and are quite sufficient for the purposes of our MIDI conversions.

3.2 Overview of midicvtpp

This document discusses how we can, with some diligence, create the INI file. It describes how we make them. Here are the broad steps:

1. In a spreadsheet, list the names and numbers of the notes or patches in both GM (General MIDI) format and in the format of your native device.
2. Convert the spreadsheet page into CSV (comma-separated values) format.
3. Use the `-csv` option to convert the CSV file into an INI file.
4. Run the `-m2m` option to convert a MIDI file into a remapped MIDI file. The `-reverse` can be used to remap in the opposite direction. However, note that often the reverse mapping will not work, since it is fairly common that the GM-to-device mapping is not one-to-one, and the key (input) value must be unique.
5. Test the new MIDI file.

A spreadsheet is used because it is a very convenient way to cut-and-paste columns, sort them in order to check them, and convert them to CSV. We use the free LibreOffice product for these purposes. It is a good product, free to use, the source-code is available, and it even handles Microsoft Office formats reasonably well.

- `MIDI-Maps.ods`.
- `GM_PSS-790_Drums.csv`.
- `GM_PSS-790_Drums.ini` and `GM_PSS-790_Drums-simple.ini`.
- `Standard-MIDI-file-format-updated.pdf`.

3.3 Usage of midicvtpp

You can run the command

```
midicvtpp --help
```

in order to see the following brief explanation. We've left off the information that applies to *midicvt*, as it is also part of *midicvtpp*. See the Overview of midicvt section.

midicvtpp adds functionality to *midicvt*:

```
 --csv-drums f   Convert a CSV (comma-separated values) file to a sectioned
                 INI drum file.  Option -o/--output specifies the full name
                 of the output file.  The default is 'out.ini', not stdout.
 --csv-patches f Convert a CSV file to a sectioned INI patch/program file.
                 Option -o/--output specifies the output name.  Default is
                 'out.ini', not stdout.
 --m2m f         Employ the given INI mapping file to convert MIDI to MIDI.

The following options require the --m2m option:

 --reverse       Reverse the mapping specified by --m2m.  Not all mappings
                 can be fully reversed; unique key values are required in
                 both directions.
 --extract n     Write only channel events from channel n, n = 1 to 16.
 --reject n      Write only channel events not from channel n.
 --testing       Only the programmer knows what this one does. :-D
```

3.3.1 midicvtpp --csv-drums

This option takes a comma-separated-value (CSV) file of drum note mappings (see The Remapping Spreadsheet) and converts it to an INI file. We use a spreadsheet because it makes it easier to lay out a bunch of notes and note names, and sort them as needed. We export the notes to a CSV file to make it easier to import the notes into an INI file. Note that sample CSV files are found in the `midicvt/tests/csvfiles` project directory.

Let's look at a couple of entries in a "drum" CSV file to be sure of what it means. Here are the first five entries from `GM_PSS-790_Drums.csv`.

```
Acoustic Bass Drum,35,N/A,35,Acoustic Bass Drum
Bass Drum 1,36,C1 Bass Drum Reverb,36,Bass Drum 1
Side Stick,37,Triangle Mute,80,Mute Triangle
Acoustic Snare,38,Synth Snare,40,Electric Snare
Hand Clap,39,Triangle Open,81,Open Triangle
```

The first line is straightforward. GM note 35 is GM drum "Acoustic Bass Drum". There is no drum note 35 for the PSS-790, so it is N/A; the drum note 35 is not likely to appear in a MIDI tune formatted for the PSS-790 anyway. Therefore, we simply map 35 to 35, no change in value. 35 is meant to produce "Acoustic Bass Drum" as the GM equivalent.

In the second line, GM note 36 is GM drum "Bass Drum 1". PSS-790 note 36 is "Bass Drum Reverb". Not the same sound, but it is close enough, so no conversion to get close to GM equivalent "Bass Drum 1".

The third line is tricky. GM note 37 is GM drum "Side Stick". The PSS-790 has no "Side Stick" sound at all. PSS-790 note 37 is "Triangle Mute". So we want to convert any drum note 37 in a PSS-790 MIDI file to the "Triangle Mute" GM note value, which is

1. It's official name in GM format is "Mute Triangle".

So let us summarize the entries in this CSV record:

1. Name of the input note in GM terms. "Side Stick".
2. Value of the input note. 37.
3. Name of the closest PSS-790 drum to approximate GM's "Side Stick". "Triangle Mute".
4. GM output note that produces the sound that approximates "Triangle Mute". 80.
5. Official GM name of the approximating GM sound. "Mute Triangle".

So, when converting a PSS-790 MIDI tune to a GM MIDI tune, drum note 37 is converted to value 80.

In the fourth line, input note 38 is the PSS-790's "Synth Snare", which is best represented by GM note 40, GM's "Electric Snare".

In the fifth line, input note 39 is the PSS-790's "Triangle Open", which is best represented by GM note 81, GM's "Open Triangle".

The CSV representation is a bit tricky. It even confused us when we came back to do some upgrading, which is why we wrote this additional material.

For example, in the `tests` directory, we ran the following command to create `GM_PSS-790_Drums.ini`:

```
$ ../midicvtpp/midicvtpp --csv-drums csvfiles/GM_PSS-790_Drums.csv
      --output inifiles/GM_PSS-790_Drums.ini
```

3.3.2 midicvtpp --csv-patches

This option takes a comma-separated-value (CSV) file of patch/program mappings (see The Remapping Spreadsheet) and converts it to an INI file.

For example, in the `tests` directory, we ran the following command to create `GM_PSS-790_Drums.ini`:

```
$ ../midicvtpp/midicvtpp --csv-Patches csvfiles/GM_PSS-790_Patches.csv
      --output inifiles/GM_PSS-790_Patches.ini
```

We concatenated `GM_PSS-790_Drum.ini` and `GM_PSS-790_Patches.ini` in order to make a combination INI file, `GM_PSS-790_Multi.ini`, the can perform both mappings.

These INI files are somewhat self-explanatory. Further explanation can be found in the developer's reference manual, *midicvt_reference_manual.pdf*.

3.3.3 midicvtpp --m2m

3.3.3.1 midicvtpp --m2m mapfile.ini

The –m2m option allows an INI file to specify how to remap between MIDI notes, program/patch numbers, and other options.

Here is an example that maps a Yamaha PSS-790's drums to General MIDI drums.

```
$ midicvtpp --m2m GM_PSS-790_Drums.ini -i stomtors-drums-16.mid
     -o stomtors-drums-10.mid
```

3.3.3.2 midicvtpp --m2m Without a Map-File

Note that one can also provide *no* mapping file:

```
$ midicvtpp --m2m -i ex1.mid -o ex1-m2m.mid
```

In this case, no remapping occurs. However, midicvtpp might *fix* the input file, as it does in this case, adding some bytes. For a detailed description of this fix, compare the following two files, found in the `tests/results` directory:

```
ex1.xxd              An annotated hex dump of the original file.
ex1-recompiled.xxd   An annotated hex dump of the "fixed" file.
```

The command shown above will yield the same results as the following commands:

```
$ midicvtpp -i ex1.mid -o ex1.asc
$ midicvtpp -c ex1.asc -o ex1-m2m.mid
```

3.3.4 midicvtpp --reverse

This option reverses the mapping specified by the –m2m option. If a MIDI file arranged for synthesizer A is remapped to an arrangement for synthesizer B, then the –reverse option can be used to remap the synthesizer B arrangement back to a synthesizer A arrangement.

However, be aware that, if the mapping from A to B isn't one-to-one, then remapping back to A will result in a file that is not quite identical to A.

3.3.5 midicvtpp --extract

The –extract option takes any events on the given channel and extracts (removes) them into the output MIDI file. Only events that have a channel parameter are extracted:

- note on
- note off
- pitchbend
- control messages

All other kinds of events pass through to the output file.

Here is an example that extracts (removes) the drum parts of a MIDI file written for the Yamaha PSS-790 consumer-level synthesizer. Note the presence of the –m2m option.

```
$ midicvtpp --m2m --extract 16 stomtors.mid stomtors-drums-16.mid
```

3.3.6 midicvtpp --reject

The –reject option is the opposite of the –extract option. It causes the events from a single channel to be dropped from the output MIDI file.

3.3.7 midicvtpp --testing

This option does whatever it does. Most likely it will cause a dump of the maps that were created from the INI file.

3.4 The Remapping Spreadsheet

Open `MIDI-Maps.ods` to follow this discussion. This document is how I keep track of some of my old equipment, and you are welcome to use it and modify it as you see fit for your purposes.

The current spreadsheet pages (tabs) are *Drums*, *Patches*, and *GM_PSS-790_Drums*, *GM_PSS-790_Patches*, and couple more tabs.

Drums. The first column shows the frequency of the corresponding MIDI keys shown in the next two columns. The "GM 1 Percussion" column shows the stock percussion names for a full General MIDI 1 drum kit.

The next columm is a natural drum-kit patch for the ZynAddSubFx/Yoshimi software synthesizers that we cobbled from a couple of other drum-kits, and allocated to all GM drum-kit notes, so that something will be heard for every note.

The "DTMF Tones" column is a patch we put together to emulate phone-dialing tones.

You can find these patches in the `contrib/yoshimi` directory of this project. *ZynAddSubFx/Yoshimi* is a complex synth, but well worth playing with, a great demonstration of completely digital synthesis, filtering, modulation, formant-processing and other techniques.

The "Yamaha PSS-790 Percussion" column lists the drum notes on our old consumer keyboard. Mapping between this synthesizer and newer equipment was our motivation for creating/enhancing the `midicvt/midicvtpp` programs.

Patches. This spreadsheet page correlates the PSS-790 program/patches number with General MIDI numbers. We haven't done anything with them yet, but just you wait!

GM_PSS-790_Drums. This is the spreadsheet page we've been moving toward for this document. It shows the drum names and notes to correlate GM1 and PSS-790 drums.

The first thing we did was to key in the basic column names and then the PSS-790 drum notes. Then we went over them, assigning the PSS-790 notes to a reasonably close GM 1 drum note.

Then we highlighted all the columns and sorted them on Column D. In doing so, we noted that some SS-790 drum notes were assigned to more than one GM note. We had to re-assign some of the notes.

Once satisfied, we could take the data on this page and save it in comma-separated value format. See the next section.

3.5 The INI File

MORE TO COME

Note that sample INI files are found in the `midicvt/tests/inifiles` project directory.

3.6 The Test Script

MORE TO COME

Note that sample files are found in the `midicvt/tests/midifiles` and `midicvt/tests/stomtors` project directories. Result files used by the test-script for comparison are found in the `midicvt/tests/results` and `midicvt/tests/results/stomtors` project directories.

4 ASCII File Format

Author(s) Chris Ahlstrom 2015-08-14

4.1 Introduction

Much of the usage documentation here comes from Piet van Oostrum, as noted elsewhere in this document.

The text representation is chosen such that it is easily recognized and manipulated by programs like *sed*, *awk*, or *perl*. Yet it is also human-readable so that it can be manipulated with an ordinary text editor.

In this way you can make changes to your midifiles using these powerful programs or even a Fortran program :=). Or you can write algorithmic compositions using a familiar programming language.

4.2 Format of the Textfile

The following representation of the MIDI events is generated. When the `-v` option is used, the longer strings shown in square brackets ("[]") are generated. The items in angle brackets ("<>") are integer or string data values

```
File header:            Mfile <format> <ntrks> <division>
Start of track:         MTrk
End of track:           TrkEnd
Note On:                On <ch> <note> <vol>
Note Off:               Off <ch> <note> <vol>
Poly Pressure:          PoPr[PolyPr] <ch> <note> <val>
Channel Pressure:       ChPr[ChanPr] <ch> <val>
Controller parameter:   Par[Param] <ch> <con> <val>
Pitch bend:             Pb <ch> <val>
Program change:         PrCh[ProgCh] <ch> <prog>
Sysex message:          SysEx <hex>
Arbitrary midi bytes:   Arb <hex>
Sequence number:        Seqnr <num>
Key signature:          KeySig <num> <manor>
Tempo:                  Tempo <num>
Time signature:         TimeSig <num>/<num> <num> <num>
SMPTE event:            SMPTE <num> <num> <num> <num> <num>
Meta text events:       Meta <texttype> <string>
Meta end of track:      Meta TrkEnd
Sequencer specific:     SeqSpec <type> <hex>
Misc meta events:       Meta <type> <hex>
```

The "<>" symbols have the following meanings:

```
<ch>                    Channel number (1 to 16).
<note>                  Note value (0 to 127).
<vol>                   Volume value (0 to 127).
<val>                   Other value (0 to 127).
<con>                   Controller number (0 to 127).
<prog>                  Program/patch number (0 to 127).
<manor>                 Minor or major flag (values are ???)
<noteval>               Either a <num> or A-G optionally followed by "#",
                        followed by <num> without intermediate spaces.
<texttype>              Text Copyright SeqName TrkName InstrName Lyric
                        Marker Cue or <type>.
<type>                  A hex number of the form 0xab.
<hex>                   A sequence of 2-digit hex numbers (without 0x)
                        separated by space.
<string>                A string between double quotes (like "text").
```

Channel numbers are 1-based; all other numbers are as they appear in the MIDI file.

"<division>" is either a positive number (giving the time resolution in clicks per quarter note) or a negative number followed by a positive number (giving SMPTE timing).

"<format>", "<ntrks>", and "<num>" are decimal numbers.

The "<num>" in the "Pb" construct is the real value (two MIDI bytes combined) In Tempo it is a long (32 bits) value. Others are in the interval 0-127

The SysEx sequence contains the leading F0 and the trailing F7.

In a string certain characters are escaped:

- Double-quotes and backslashes are escaped with a backslash.
- A zero byte is written as "\\0"
- CR and LF are written as "\\r" and "\\n" respectively
- Other non-printable characters are written as "\\x<2 hex digits>"
- When "-f" is given long strings and long hex sequences are folded by inserting "\\<newline><tab>". If in a string the next character would be a space or tab it will be escaped by a backslash.

This facility is for those programs that have a limited buffer length. Of course parsing is more difficult with this option (see below).

4.3 Input

In the midi2text project, the processing was split between two applications. We'll use that convention in this usage description. Maybe we'll make UNIX shortcuts for them.

mf2t is the same as *midicvt –input*. This command converts MIDI files to ASCII text files.

t2mf is the same as *midicvt –compile*. This command converts the ASCII text file back to a MIDI file.

"t2mf" will accept all formats that "mf2t" can produce, plus a number of others (to be determined).

Input is case insensitive (except in strings) and extra tabs and spaces are allowed. Newlines are required but empty lines are allowed. Comment starts with "#" at the beginning of a lexical item and continues to the end of the line. The only other places where a "#" is legal are inside strings and as a sharp symbol in a verbose symbolic note.

In symbolic notes "+" and "#" are allowed for sharp notes, and "b" and "-" for flat.

In the "bar:beat:click" time construct, the ":" may also be "/".

On input, a string may also contain "\t" for a tab, and in a folded string any whitespace at the beginning of a continuation line is skipped.

Hex sequences may be input without intervening spaces if each byte is given as exactly 2 hex digits. Hex sequences may be given where a string is required and vice versa.

Hex numbers of the form "0xaaa" and decimal numbers are equivalent. Also allowed as numbers are "bank numbers" of the form "'123". In fact this is equivalent to the octal number 012 (subtract 1 from each digit, digits 1-8 allowed). The letters "a-h" may also be used for "1-8".

The input is checked for correctness but not extensively. An error message will generally imply that the resulting MIDI file is illegal.

4.4 Useful Hints

Channel number can be recognized by the regular expression "/ch=/". Note numbers can be recognized by "/n=/" or "/note=/", program numbers by "/p=/" or "/prog=/". Meta events can be recognized by "/^Meta/" or "/^SeqSpec/". Text events can be recognized by /"/, continued lines by "/\$/", continuation lines by "/$/" (the tab character).

In *awk* each parameter is a field, in *perl* you can use split to get the parameters (except for strings).

The following Perl script changes note off messages to note on with vol = 0, deletes controller 3 changes, makes some note reassignments on channel 10, and changes the channel numbers from channel 1 depending on the track number.

`test.pl`:

```
%drum = (62, 36, 63, 47, 65, 61, 67, 40, 68, 54);

while (<>) {
   next if /c=3/;
   s/Off(.*)v=[0-9]* /On\1v=0/;
   if (/ch=10/ && /n=([0-9]*)/ && $drum{$1}) {
      s/n=$1/"n=".$drum{$1}/e;
   }
   if (/^MTrk/) {++$trknr ; print "track $trknr\n";}
   if ($trknr > 2) { s/ch=1\b/ch=3/; }
   else  { s/ch=1\b/ch=4/; }
   print || die "Error: $!\n";
}
```

See the actual file for the best results.

This is the corresponding Awk script.

`test.awk`:

```
BEGIN { drum[62] = 36; drum[63] = 47; drum[65] = 61; \
   drum[67] = 40; drum[68] = 54 }
/c=3/ { next }
($2 == "Off") { $2 = "On"; $5 = "v=0" }
/ch=10/ { n = substr($4, 3); if (n in drum) $4 = "n=" drum[n] }
/^MTrk/ { trknr++ }
/ch=1 / { if (trknr > 2) { $3 = "ch=2" } else { $3 = "ch=3" } }
{ print }
```

Good luck!

```
   $Id: readme.,v 1.6 1995/09/23 22:27:48 piet Rel $
```

5 Licenses, MIDIcvt Projects.

Author(s) Chris Ahlstrom 2015-08-14

5.1 License Terms for the midicvt project.

Wherever the tag $XPC_SUITE_GPL_LICENSE$ appears, substitute the appropriate license text, depending on whether the project is a library, application, documentation, or server software. We're not going to include paragraphs of licensing information in every module; you are responsible for coming here to read the licensing information.

These licenses apply to each sub-project and file artifact in the **XPC** library suite.

Wherever the term **XPC** is encountered in this project, it refers to the **XPC Development Suite** on **SourceForge.net**, which is also called the **XPC Library Suite**, and may be provided at other sites.

5.2 XPC Application License

The **XPC** application license is the **GNU GPLv3**.

Copyright (C) 2008-2014 by Chris Ahlstrom

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU GPL version 3 license can also be found here:

http://www.gnu.org/licenses/gpl-3.0.txt

5.3 XPC Library License

The **XPC** library license is the **GNU LGPLv3**.

Copyright (C) 2008-2014 by Chris Ahlstrom

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Lesser Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU LGPL version 3 license can also be found here:

http://www.gnu.org/licenses/lgpl-3.0.txt

5.4 XPC Documentation License

The **XPC** documentation license is the **GNU FDLv1.3**.

5.5 XPC Affero License

The **XPC** "Affero" license is the **GNU AGPLv3**.

At the present time, no **XPC** project uses the "Affero" license.

5.6 XPC License Summary

Include one of these licenses in your Doxygen documentation with one of the following Doxygen tags:

```
\ref midicvt_suite_license_subproject
\ref midicvt_suite_license_application
\ref midicvt_suite_license_library
\ref midicvt_suite_license_documentation
\ref midicvt_suite_license_affero
```

For more information on navigating GNU licensing, see this page:

```
http://www.gnu.org/licenses/
```

Copies of these licenses (and some logos) are provided in the `licenses` directory of the main project (or you can search for them at *gnu.org*).