GNMIDI
MIDI TOOLS for Windows

GNMIDI

A software for MIDI friends

by Günter Nagler

MIDI is the language that most electronic musical instruments, computers and recording studios have in common.

A MIDI file tells the playing device all the steps that the synthesizer must do to produce a song instead of only sound.

GNMIDI gives you the opportunity to join in the fun that musicians have with the use of MIDI.

Don't be afraid that working with MIDI is too difficult or requires too much knowledge of music, techniques or computers. With GNMIDI it's easy and fun to work with MIDI files.

GNMIDI is very efficient. It is small enough to put on a floppy disk and take with you anywhere. It will even run right from the disk. No installation necessary!

GNMIDI - MIDI tools for Windows

Printed: April 2025 in Austria, Graz

Programming

Günter Nagler, Austria

Special thanks to:

All the people who helped that GNMIDI became a successful software idea, the users who sent new ideas and helped to test them.

All the professional and amateur musicians who sent me important input about what musicians really need.

All the users who talk about GNMIDI and recommended it to new users.

Genea, who spent valuable time in proof-reading this document.

Microsoft for standardizing MIDI drivers and making it possible that users can use MIDI on different systems .

KORG who developed the wonderful i-series MIDI keyboards that I am using.

YAMAHA who developed the fascinating CVP-305 digital piano that I am using.

Share-It for their superb online order services.

Table of Contents

1 Introducing GNMIDI

1.1 Why GNMIDI?

GNMIDI Software offers a huge list of operations that help preparing MIDI files for musicians. It is easier to use than a sequencer and is an ideal supplement to MIDI sequencer software. Many important operations that are difficult to do with a MIDI sequencer can be done within GNMIDI with few button clicks.

GNMIDI features are useful in following areas:

- Karaoke Playing and Editing
- Lyrics synchronization
- Lyrics conversion for many keyboard displays
- Sysex Dump
- MIDI Archive
- MIDI file modification
- Entertainment
- Mobile phone melodies
- MIDI Recording
- MIDI Playing
- quick MIDI file information display
- Chord analysation
- Format conversions
- Check and Repair
- Compression
- Copy parts from a MIDI file
- View into MIDI file internals

GNMIDI contains also a few features for mp3 song files, but is mainly for MIDI users:

- enter mp3 song text
- synchronize mp3 song text line by line (Karaoke)
- play mp3 song with karaoke text
- find mp3 song files in archive by words
- remember additional information to mp3 song files in archive (rating, comment, ...)
- entertainment player plays mp3 song files in random order considering files that contains certain words, rating, playing frequency

1.2 About the GNMIDI project

Project author: Günter Nagler, Austria

From 1995-1999 I wrote many command line MIDI utilities for DOS. This program should offers many operations of these utilities and more. Many people are not familiar with computers using DOS or UNIX command lines, so **GNMIDI** - **for Windows** - is much easier to use.

The project started with some popular utilities and I may add new ones from time to time. The project has become a full featured program and is in a very stable state.

The software is available in **English** or **German**.

GNMIDI is currently tested under following environments: **Windows 7, 8, 10**

GNMIDI might work with Windows systems Win95, WinME, Win98, WinNT4, Win2000, Vista,

Windows XP but this is not listed in supported systems because we currently do not test on these environments. We can not give any guarantees for these systems that all functions of a GNMIDI version will work for these systems. Test the demo carefully on your system to be sure that it works well. Please let us know your results if you are using one of these operating systems.

1.3 End user license agreement

GNMIDI (c) 1997 was created by Günter Nagler.

GNMIDI 3.0 and later is not free, it is commercial. You may try it free for 14 days to test it and then you must purchase a license if you want to use it further or delete the program.

Prices may change. Download newest demo version for current price.

Purchasing a GNMIDI license is only obtaining the permission to use GNMIDI software. It is not allowed to resell or share a license, only the registered owner listed in the license file has permission to use GNMIDI.

The end user license agreement that is viewed at start of the program and can be downloaded from https://www.gnmidi.com GNMIDI product page.

Demo version of the software asks you at program start if you are willing to accept our software usage conditions and copyright rules. You can quit immediately and delete the software from your computer if you don't agree. If you enter the software you agree to the conditions.

1.4 Installation

This program runs under Microsoft Windows® versions **Windows 7, 8, 10** and **11.** If S-Mode is active in Windows Home then Windows does not allow to install or start the application**.**. It does not run under Windows 3.x and older. It is currently not tested with other Windows versions like **WinME, Win95, Win98, Win-NT, Win2000, Win XP, Vista,** or **WINE emulator** (Linux, MacOS) therefore we cannot guarantee that it works with them. If you need to use this software with those unsupported systems please try demo self and register with knowledge that we maybe cannot fix system depending software problems. You may try to use GNMIDI with these systems self without guarantee.

if you download MSI file for installation the program will be installed into program files (x86) and create a shortcut on desktop and app menu.
Deinstallation is done using Windows deinstall program (search for deinstall in Windows run) and click the line with the wanted program name.
The gnmidi.ini file in your documents folder must be deleted manually.

If you download ZIP file for installation you need to extract the content of the ZIP archive into a new empty GNMIDI folder (e.g. C:\GNMIDI).
Create a shortcut on the desktop to the program gnmidi.exe (green symbol) with Explorer. Alternatively start the program with Windows Explorer in the GNMIDI folder.
Deinstallation is done by deleting the GNMIDI files in your GNMIDI folder and the gnmidi.ini file in your documents folder.

Install your personal license as explained in the mail where it was sent. Ask the developer if there are difficulties.
After correct installation of license the program should display your name at bottom side of the program (registered by ...).

The personal license file *.gnlic might be in application folder

C:\Users\<username>\AppData\Roaming\GN MIDI Solutions\GNMIDI or in GNMIDI program folder.

At first program start GNMIDI creates a new readable file **gnmidi.ini in your personal *My Documents* folder**, which remembers your recent program settings.

1.5 License

GNMIDI 3.0 and later is not Freeware it is commercial software (that you can try as demo before buying).

A purchased license is valid for GNMIDI 3 version which was released during your order and GNMIDI 3 update versions released within 2 years.

Installed license will turn the demo into full version. The demo limitations will be away. The end user license agreement must be agreed at first program start.
The name of the user who has permission to use the full software will be displayed at bottom side of the program (registered by ...).

There are two kind of licenses for GNMIDI available: **Professional and Light**

The Light license is for users who only need a smaller part of the operations and therefore pay a smaller license fee. Upgrading from light to Professional is possible.

Differences between Professional and Light license:

Operation	Professional (GNMIDI3)	Light (GNMidiLight3)
Batch operations	yes	no
Chords Editor	yes	no
Karaoke Editor	yes	no
Playlists	yes	no
Price	higher	inexpensive
Software versions	GNMIDI 3.1 - 3.99	GNMIDI 3.13 - 3.99
Updates	2 years free updates included	2 years free updates included

GNMIDI Light license (GNMidiLight3) can only be activated with program versions GNMIDI 3.13 or higher. GNMIDI 3.1 - 3.12 can only be activated using GNMIDI Professional license (GNMIDI3).
Please notify that program versions newer than 2 years after your order date can not be used free with your license (requires upgrade).

Software production takes much time, so please **register after testing (14 days) if you like it**. The concept makes it safe for you, you **don't purchase something that you haven't tried**.

License will be delivered by email (no cdrom included).

Important: GNMIDI 2 license files are not accepted by GNMIDI 3. Please check if there is an upgrade possibility.

Online order of a software license:
https://www.gnmidi.com/gnorderen.htm

Upgrading from Light to Professional License or **advancing updates by additional 2 years:**
Please contact info@gnmidi.com by email or use our contact form at https://www.gnmidi.com

1.6 How to register GNMIDI

A GNMIDI registration is license contract between user and software author that permits the user to use the software without demo limitations. See End user license agreement that the user must agree before using the software.

GNMIDI is available in **English and German language**.

Order a software license online:
https://www.gnmidi.com/gnorderen.htm

Program and updates can be downloaded from
https://www.gnmidi.com/download (Austria)

1.7 How to print this help file

Best printing of these help pages can be done using the PDF version of this help.
You can download this document from
https://www.gnmidi.com/gnmidien3.pdf

In order to view PDF files, the free Acrobat Reader from Adobe is required.

Delivery does not contain a printed manual.

1.8 GNMIDI.INI

gnmidi.ini is a text file in your **My Documents** folder which stores your personal program settings. This file will be created automatically at first program start. Most settings are done automatically by program.
Some very rarely used settings can only be modified by editing the GNMIDI.INI file with Notepad text editor, this will be mentioned by the relating documentation to an operation.

Most settings can be found in section "Settings" after the line:

```
[Settings]
MidiMode=1
RecordDevice=0
MidiFollowLyrics=1
MidiAdjustVolume=0
MidiCommonVolume=1000000
MidiAllCommonVolume=1000000
MidiAutoCompression=0
TextFontAttrib=-13 0 0 0 700 0 0 0 0 3 2 1 18
TextFontName=Perpetua
KaraokeFontAttrib=-19 0 0 0 700 0 0 0 0 3 2 1 49
KaraokeFontName=Courier New
...
```

1.9 Choose a text editor

`[menu settings]`

GNMIDI will write some results in text files and display them with the Windows text editor Notepad. This editor can not handle very big files.

Alternatives

There exist text editors which are compatible with Notepad and have more facilities than Notepad and less limitations
e.g: Notepad2 (Freeware), Notepad++
In internet you could find descriptions how to replace Windows Notepad.exe completely against some other trustable software.

GNMIDI.INI setting to use an other text editor than Notepad:

```
[Settings]
TextEditor=c:\utils\notepad2.exe
```

In menu settings a dialog to choose a text editor can be found.
It suggests notepad2.exe or notepad++.exe and tries to find them if installed.
Or you can use a dialog edit field to enter a text editor .exe file path or click ... button to select the application manually.

Hint: office products can not be used for this purpose. It must display and edit .txt file (ansi or UTF-8 text format).

1.10 GNMIDI support

GNMIDI homepage

https://www.gnmidi.com (Austria)

the homepage contains newest GNMIDI demo.

Within 2 years registered users may **update to a newer GNMIDI version** without costs:
Download newest GNMIDI demo in chosen language and install it into your current GNMIDI folder or into a new folder.

Questions about GNMIDI
email: **info@gnmidi.com**
phone: there is no phone support available.

Bug reports, comments, suggestions, spelling errors in software and help manual are welcome.

Support for registered users
email: support@gnmidi.com

Report abuses
email: abuse@gnmidi.com

2 GNMIDI user interface

2.1 The principles of using GNMIDI

Operations
GNMIDI consist of many operations. Some can be applied any time and most others must be applied to an open MIDI document.
The operations can be found in the menus and some important have icons on a toolbar. The menu or toolbar item is grayed if the operation is currently not available. Use File Open operation to load a MIDI file.

Document windows
Most GNMIDI operations don't modify your original files. Every time when you apply an operation to a MIDI file document then a new temporary file will be created in a temporary folder and opened in a new window. It is necessary to save the file after testing the result (by playing the file). Click on a window caption to activate the document, the window will get into front. You can close unnecessary windows of unused intermediate results when a work is done.

Drag and Drop
GNMIDI can open MIDI files by drag and drop method. Open Windows Explorer and GNMIDI and select MIDI files in Windows Explorer and drag them with pressed left mouse button into the GNMIDI application and drop them (release left mouse button).

Menus
The menu at top of application offers to start available commands. Use the underlined menu shortcut keys for quicker use of a command.

Dialogs
Many operations require parameters that will be offered in a dialog and they must be entered so that GNMIDI will do exactly what you want. When a dialog appears you should look at the current parameter settings and modify them if necessary. Use the mouse buttons to click on the dialog controls and use keyboard to enter text into edit boxes. At end you can accept the parameters and start the operation with OK button or you can abort the dialog and operation with Cancel button.

Tool bars
Toolbars contain many symbols that can be clicked with left mouse button. Move the mouse over a symbol to display a tool tip that tells what it does and show a description in status bar. Toolbars can be moved to other positions.

Status bar
Status bar shows important messages on the left side and tells your name if you are using the registered software.

Hot keys
Some important operations can be started with a key combination. The menu entry shows the key code near the operation name. For key combinations you need to press more keys at same time (usually Alt or Ctrl or Shift key together with a letter key).

F1	Help
Space	Play or stop playing
Ctrl-Space	Play with standard MIDI player
Ctrl-A	Start entertainment player
Ctrl-I	Player status information
Ctrl-K	Karaoke editor
Ctrl-Z	Synchronizing editor

Ctrl-R	MIDI Recorder
Alt-X	Sysex transfer
Ctrl-O	Open MIDI file
Ctrl+Shift+S	Save MIDI file to original file (with backup created)
Ctrl-S	Save MIDI file to a new file
Ctrl-E	Edit text
Alt-F4	Exit
Ctrl-F4	Close MIDI file

Hint: Ctrl key might be Strg key on your keyboard

Batch conversion

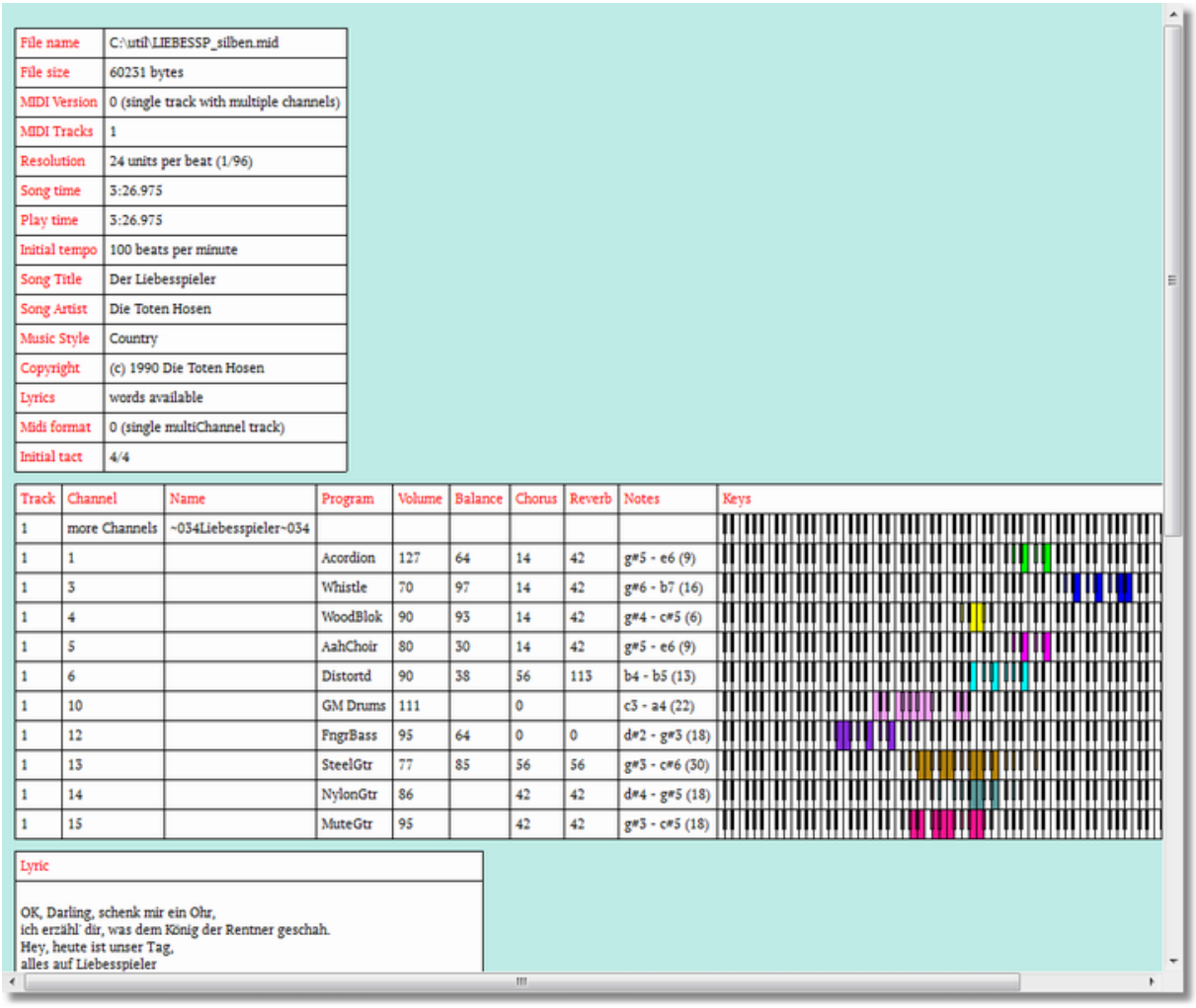
Some important operations can be applied to a folder full of MIDI files. The operations are listed in File menu under Batch conversion. The results are stored in a destination folder using same file names.
Using GNMIDI Light license batch operations are not available.

Backup original MIDI files

Many operations are not reversible. Keep backup of original MIDI files before you are overwriting your original files!

2.2 Information displayed in a MIDI document window

The upper part of a MIDI document window contains following information if a standard MIDI file or a MP3 file is loaded:



File name	C:\util\LIEBESSP_silben.mid
File size	60231 bytes
MIDI Version	0 (single track with multiple channels)
MIDI Tracks	1
Resolution	24 units per beat (1/96)
Song time	3:26.975
Play time	3:26.975
Initial tempo	100 beats per minute
Song Title	Der Liebesspieler
Song Artist	Die Toten Hosen
Music Style	Country
Copyright	(c) 1990 Die Toten Hosen
Lyrics	words available
Midi format	0 (single multiChannel track)
Initial tact	4/4

Track	Channel	Name	Program	Volume	Balance	Chorus	Reverb	Notes	Keys
1	more Channels	~034Liebesspieler~034							
1	1		Acordion	127	64	14	42	g#5 - e6 (9)	
1	3		Whistle	70	97	14	42	g#6 - b7 (16)	
1	4		WoodBlok	90	93	14	42	g#4 - c#5 (6)	
1	5		AahChoir	80	30	14	42	g#5 - e6 (9)	
1	6		Distortd	90	38	56	113	b4 - b5 (13)	
1	10		GM Drums	111		0		c3 - a4 (22)	
1	12		FngrBass	95	64	0	0	d#2 - g#3 (18)	
1	13		SteelGtr	77	85	56	56	g#3 - c#6 (30)	
1	14		NylonGtr	86		42	42	d#4 - g#5 (18)	
1	15		MuteGtr	95		42	42	g#3 - c#5 (18)	

Lyric

OK, Darling, schenk mir ein Ohr,
ich erzähl' dir, was dem König der Rentner geschah.
Hey, heute ist unser Tag,
alles auf Liebesspieler

Here is the explanation of fields:

Filename:
the full pathname of the MIDI file, it is a path into the Windows temporary folder and filename starts with new00 if the file is temporary (not saved yet).

Filesize:
the number of bytes in this file

MIDI Version:
the version (also called MIDI format or MIDI type) is a number between 0 and 2.
Version 0 MIDI file has only one track that contains all MIDI commands of all channels sorted by time.
Version 1 MIDI file has more tracks, where first track is reserved for conductor and all other tracks should only contain one channel.
Version 2 MIDI file has more tracks, where each track is a new song.

MIDI tracks:
The number of tracks in the MIDI file. Each track can contain MIDI commands.

MIDI resolution:
The resolution tells about the smallest possible note and pause steps in the song.
24 units per beat means that the smallest note length would be a 1/96 note.

Song time:

The total time of the song in minutes:seconds:milliseconds (from beginning till last end of track)

Play time:
The total play time of the song in minutes:seconds:milliseconds (from beginning till last event in the song)

MIDI initial tempo:
The tempo that is used at start of the song, it is 120 beats per minute if no tempo is specified by the song.

MIDI copyright:
Copyright text of the MIDI publisher.

MIDI lyrics:
tells if lyrics are inside the song and which format it uses (karaoke words, lyrics, other).

Track list:
The track list displays a table for a MIDI song with columns track number, channel number, track title, volume, blance, chorus, reverb and a keyboard view that shows the notes used in song by each channel.
During playing a MIDI song the keyboard view displays which note keys are currently played by each channel.

Hint: Some table entries can be clicked with left mouse button to start a function that allow to modify the information (e.g. click into column Volume starts the set controller value operation with the channel in this table row and controller type Volume).

Window settings can extend this short track info to verbose track info, which tells also other initial parameters that are used in this track.

Text:
Lyric:
Marker:
Other:
The song can contain text information of different kind. This information is collected here for each of these types.

More fields are shown when extra information is added by the user e.g. artist, song title, melody channel, transpose value, rating, comment.

The buttons Full/Middle/Karaoke may be used to move the splitter of this window to a position that the upper information part is fully visible or the bottom karaoke part is fully visible or both parts are visible. There are also options in menu window available.

There is an option in menu settings to show verbose track information.

The text can be copied to clipboard (Ctrl-C) or to a text editor (Ctrl-E).

The MIDI document window contains following information if a text file generated by MIDI to ASCII text conversion is loaded:

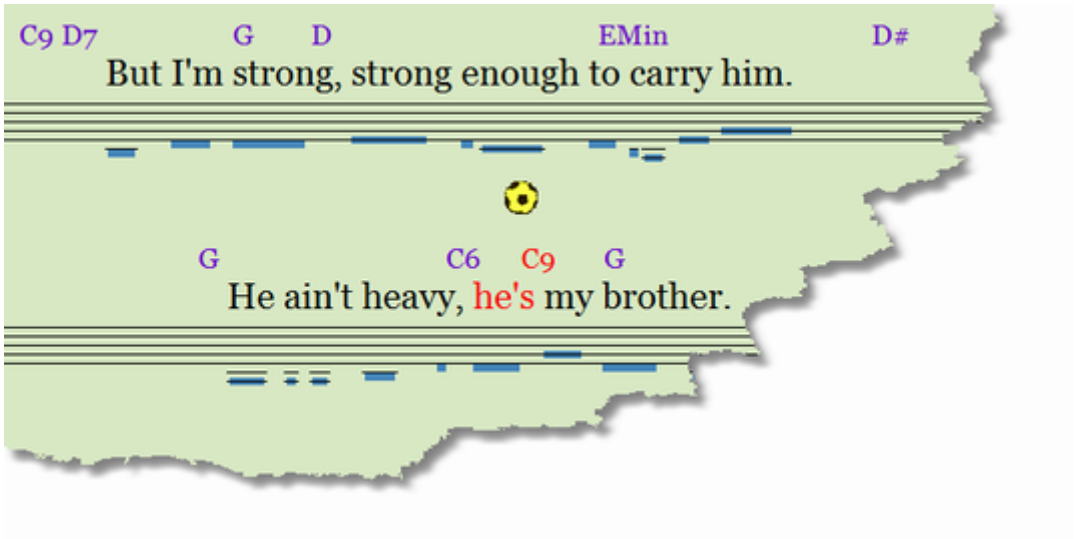
```
// C:\gnmixer\liebessp.mid
mthd
  version 1 // several tracks with separated channels to play all at once
  // 13 tracks
  unit 24 // is 1/4
end mthd
```

```
mtrk  // track 1
 /* U0 */  /* 0ms */   trackname "Liebesspieler (Die Toten Hosen)"
 /* U0 */  /* 0ms */   copyright "(c) 1990 Die Toten Hosen"
 /* U0 */  /* 0ms */   marker "sequenced 1998 by GN"
 /* U0 */  /* 0ms */   tact 4 / 4 24 8
 /* U0 */  /* 0ms */   beats 100.00000 /* 600000 microsec/beat */
  8279; /* U8279 */  /* 206975ms */
end mtrk

. . .
```

Play list

the play list box displays the file name of the songs. Use the shortkeys for playlist to select the next item. Click on the file name if you want to open the song in a new window.

2.3 Karaoke display



Load a karaoke file (.kar, MIDI with text or lyrics or markers) and play the song with internal player. The bottom half of the MIDI document window displays the karaoke song text. During playing the position of the current song text is moved into visible area and the current syllable/word/chord is colored.

Hint:
Use the tool buttons INFO, INFO/KAR, KAR to switch between displaying information view and karaoke view or both splitted.

Hint:
you can maximize the karaoke display area by **clicking left mouse button taste when you are over the black area on the splitter** (cursor will change when your mouse is over this part).

MOUSECLICK	switch views in order: INFO => INFO/KAR => KAR => INFO
SHIFT+MOUSECLICK	switch views in order: KAR => INFO/KAR => INFO => KAR

Hint:
You can change the karaoke text font in menu Window and view syllable breaks (-) in menu

Settings.

Hint:
Chords of some known chord formats are displayed a line above song text.

Hint:
If chords should never be displayed above lyrics, they can be turned off in text file GNMIDI.ini with following setting:

```
[Settings]
SupportChordDisplay=0
```

Hint:
Use themes dialog in menu window to choose a combination of karaoke font, karaoke ball type, text and background colors and background image. You can copy an existing standard theme and define your own theme.

Hint:
Some chord formats use special characters to identify chords e.g. [Cm] {Cm} (Cm) %Cm "Cm". During karaoke lyrics display this special characters are not displayed (only Cm will be displayed). use following setting to display the original chord text:

```
[Settings]
DisplayOriginalChordFormat=1
```

Hint:
in menu Settings there is an option to turn on/off display of a bouncing ball (during karaoke display the ball is bouncing from syllable to syllable)
Warning: This option requires much computer power. **Do not use this option during live performance**.

Hint: In menu settings filling longer MIDI lyrics pauses by beat counting can be turned on/off.
Use following setting to change the displayed text (default is *) for each counted beat:

```
[Settings]
KaraokeFillLongerPausesText=*
```

Hint:
The score lines with melody note positions can be turned on/off in menu settings.
If scorelines are displayed melody note names can be displayed optionally with a setting.
Choose one or more melody channels using edit description dialog (right mouse click into karaoke view) that the correct notes are displayed. Use value 0 for no melody channel.
Score lines are only available for MIDI songs that contain notes on the chosen melody channel.
Very deep and very high notes are automatically moved by octaves near to scorelines.
Raised semitones (#) will be colored different at top. Reduced semitones (b) will be colored different at bottom.
The colors of the note positions can be changed in GNMIDI.ini settings:

```
[Settings]
KaraokeScorenoteColorFlatNormal=FFF000
KaraokeScorenoteColorFlatActive=FFFF00
KaraokeScorenoteColorNormal=4486BB
KaraokeScorenoteColorActive=F2800D
```

By default the size of score lines are related with the toolbar size setting in menu. With ini setting the line thickness in pixel can be set directly (1-5).

```
[Settings]
KaraokeScorelineThickness=1
```

Hint:
turn on/off settings show marker text to display or hide marker lines inserted into karaoke view
The colors can be defined in gnmidi.ini settings by adding lines

```
[Settings]
MarkerTextColor=9D8600
MarkerBackColor=FFF000
```

2.4 Short keys

<space>	Play/Stop start or stop playing current song
Alt+<space>	Pause/Continue
Ctrl+<space>	start song with external player
Shift+TAB	Player jump 15 seconds back
TAB	Player jump 15 seconds forward
Ctrl+Arrow-Links	select previous song in play list
Ctrl+Arrow-Rechts	select next song in play list
Shift+<space>	play next (selected) song in play list
Alt+E	set end of loop in player
Alt+L	set begin of loop in player
Ctrl+I	Player status dialog
Ctrl+O	File open
F3	Open favorites
PLUS	Add current document to favorites
MINUS	Remove current document from favorites
Alt+PLUS	add more files to favorites
Shift+Ctrl+S	File save (same name and folder as original file)
Ctrl+S	File save as (new name or other folder)
Alt+B	Batch operation menu
Ctrl+B	open folder
Alt+1,Alt+2,Alt+3,Alt+4,Alt+5	start external tool (command defined in gnmidi.ini setting)
Ctrl+A	Autoplay
Ctrl+EINFG, Ctrl+C	copy song information or selected text to clipboard
Shift+Ctrl+C	chords editor
Ctrl+Alt+C	MIDI cripple
Ctrl+N	create new empty MIDI file
Ctrl+X, Shift+ENTF	cut selected text to clipboard
Shift+EINFG, Ctrl+V	insert text from clipboard into text
Ctrl+E	open song information or ASCII text in a text editor
Ctrl+Alt+F	MIDI filter dialog for deleting certain kind of MIDI commands
Ctrl+F	search text in song files
Shift+Ctrl+F	find repeating parts (patterns) in MIDI files
Shift+Alt+F	convert chords format
Ctrl+Alt+P	printing preview
Shift+Ctrl+Alt+P	printer settings
Ctrl+Alt+	S clean MIDI commands
Ctrl+K	karaoke editor
Ctrl+M	MIDI map
Ctrl+Alt+I	MIDI description
Ctrl+T	edit MIDI track titles
Ctrl+Alt+T	edit MIDI META text commands
Ctrl+P	calculate maximum polyphony (how many notes are playing at same time)
Ctrl+R	MIDI recorder
Ctrl+Z	synchronisation editor
Alt+X	Sysex dump (send and receive sysex commands)
F6	next window
Shift+F6	previous window
F9	move play list to foreground

2.5 Menu

Following items are in the menu bar at top:

Alt-F File menu

Alt-A Analyse menu

Alt-C Convert menu

Alt-M Modify menu

Alt-P Player menu

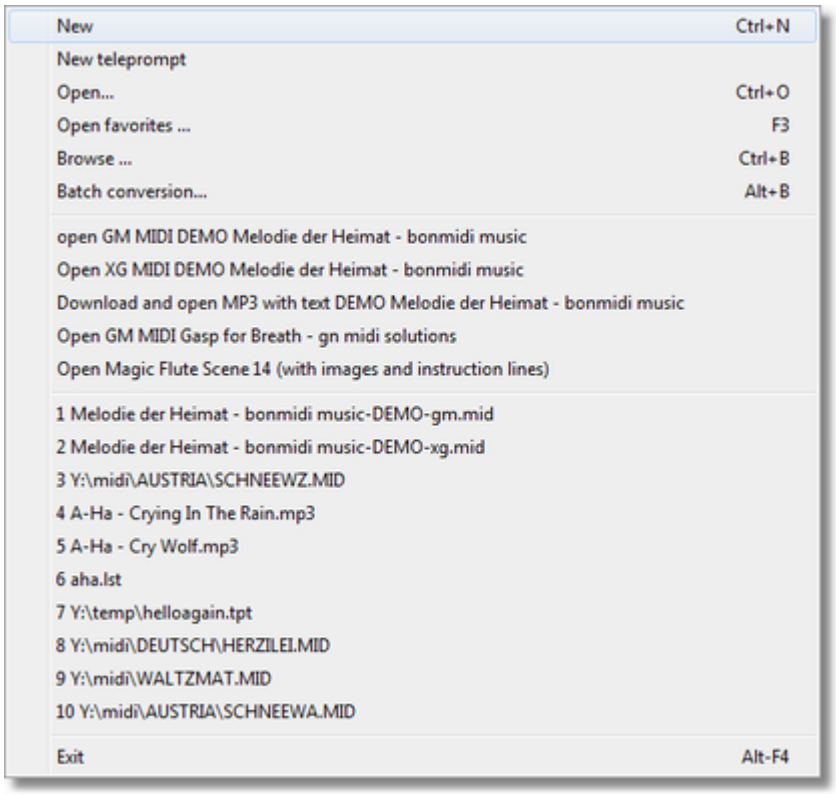
Alt-S Settings menu

Alt-W Window menu

Alt-H Help menu

The main items can also be opened by keystrokes Alt+underlined character.
The submenu items can be selected by the up and down arrow keys or by pressing the underlined character in the menu item text.
A popup menu item ▸ can also be opened by right arrow key.
Some menu items contain a key shortcut in its menu title, this key combination can be used to start the operation directly.

2.5.1 File menu



New
Create a new empty MIDI file

New teleprompt ...
creates a selfscrolling text for assisting performance

Open...
Open an existing MIDI file.

Open favorites
open files from your favorites list, add more files, remove files

Add current file to favorites list
insert or remove current document file in favorites list

Browse ...
Display folder content

Edit
open a text editor for exporting MIDI information or to edit a MIDI file that is converted to ASCII text.

MIDI song description
show and modify song description (author, title, rating, style etc.)

Close
Close active MIDI information window and stop MIDI song if this song is currently playing.

Save
Save modified song and overwrite the file. A backup file of the original song is created in the backup folder.

 Save As ...
Save or rename song to a new filename and location

Delete
Erase current MIDI file

 **Copy information**
copy MIDI information into clipboard

 Print Lyrics
print song text

Batch conversion
apply operations to whole MIDI folder. Using GNMIDI Light license batch operations are not available.

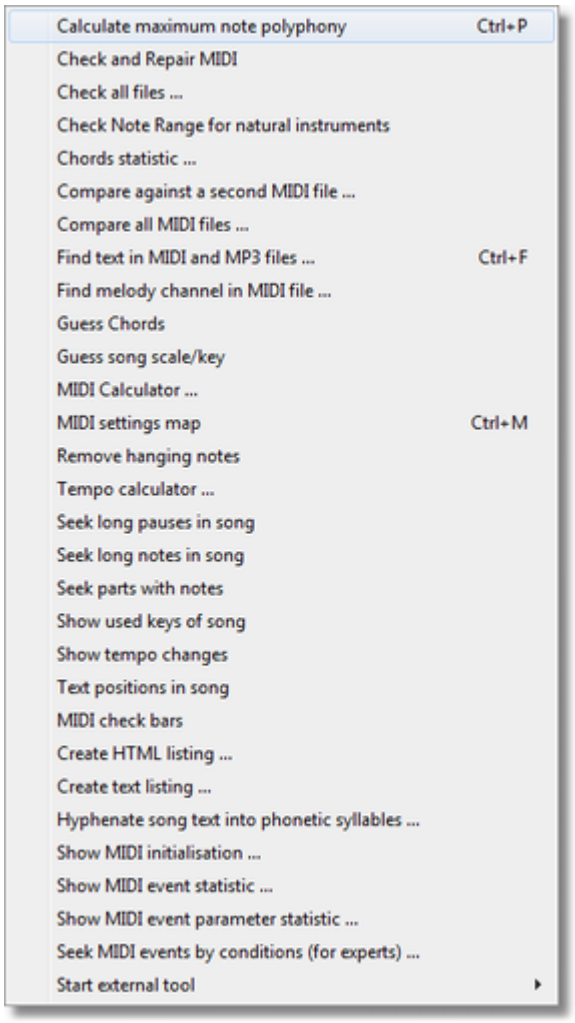
Open Demo music files

Exit
Exit
close the application

Hint: The backup folder is by default inside the program application data folder. In gnmidi.ini settings file the backupfolder can be changed to an other path (it must be writable):

```
[Settings]
backupfolder=c:\gnmidibackupfolder
```

2.5.2 Analyse menu



Calculate maximum note polyphony
Calculates maximum number of notes playing at same time in a MIDI song

Check and Repair MIDI
Checks if the MIDI file is standard MIDI compatible (.mid) and tries to repair corrupt MIDI files.

Check all MIDI files ...
Check validity of all MIDI files in a folder

Check Note range for natural instruments
check for some sound instruments if notes are used in a realistic range

Compare against a second MIDI file
shows differences between two similar MIDI files

Compare all MIDI files
compare MIDI files in two folders to find identical or similar MIDI files

 Find text in MIDI and MP3 files ...
search text in MIDI files

Find melody channel in MIDI file ...

 Guess Chords
chord analysation for whole song

 Guess song scale/key
calculate song key for a score sheet

 MIDI Calculator ...
Calculate MIDI song positions

 MIDI settings map
song parameters landscape

 Remove hanging notes
stop notes that are not stopped and play forever

 Tempo calculator ...
calculate bpm tempo of a song by clicking to the song rhythm

Seek long pauses in song
finds positions in MIDI song where longer pauses start

Seek long notes in song
finds notes in MIDI song with long durations

Seek parts with notes
seeks in all channels for parts that play notes

 Show used keys of song
display song key changes used in MIDI song.

Show tempo changes
displays positions and tempo of all tempo changes in the MIDI song.

Text positions in song
displays text information contained in the MIDI song and their positions

MIDI check bars
checks if bars of a MIDI song are shorter than expected

Create HTML listing ...
creates a HTML text for current song or play list containing information about the content

Create text listing ...
creates a text for current song or play list containing information about the content

Hyphenate song text into phonetic syllables ...
splits English or German song text automatically into syllables as they are problably sung

Show MIDI initialisation ...
shows the initialising MIDI commands before first note starts

Show MIDI event statistic ...
Shows tables of used MIDI event counts within a song

Show MIDI event parameter statistic
Shows tables with parameter value ranges of used commands

Seek MIDI events by conditions (for experts) ...
use script commands to find MIDI events with matching criterias

Start external tool
Sub menu allows to start other applications with optional current GNMIDI document path

2.5.3 Convert menu

0 1 Change MIDI format
Change MIDI format 0 to 1 or reverse

RIFF MIDI to Standard MIDI
Extract MIDI file from a .rmi file

Generate MIDI Medley
Combine MIDI files to one MIDI song (medley).

Split MIDI medley
splits a medley song generated by GNMIDI into parts

Karaoke to MIDI with lyrics
Converts MIDI files that contain song text to MIDI formats that can be displayed with certain keyboard displays and software.

MIDI with lyrics to Karaoke (.kar)
Converts MIDI files that contain song text to .kar format

Compress MIDI
Decreases MIDI file size without changing the musical content.

Decompress MIDI

stores the MIDI file without using compression (for older devices)

Convert MIDI to favorite text format
convert binary MIDI file to text using the selected text converter in menu settings

MIDI to ASCII text
Converts binary MIDI file to a readable text

MIDI to readable list
converts binary MIDI file to a text list with a line per MIDI event sorted by time

MIDI to bar separated list
converts binary MIDI file to a bar separated list with bar sections that contain MIDI events that in this bar

ASCII text back to MIDI
Converts modified text generated by Convert MIDI to favorite text format operation back to MIDI file (detects used text converter from text)

Reverse MIDI song (mirror)
Generates a MIDI song that allows to play a MIDI song reverse (from end to beginning)

MIDI to RTTTL (ring tones) ...
convert MIDI melody to a phone ring tone in RTTTL format

RTTTL to MIDI
convert RTTTL phone ring tones into a MIDI file.

MIDI to Parsons code ...
convert MIDI melody to Parsons code (for identifying a melody)

Initialize GM/GS/XG/GM2 mode
adds a reset mode sysex command or converts MIDI file to General MIDI.

Convert MIDI to (.csv) speadsheet
converts MIDI data into a table that can be viewed and modified using spreadsheet applications

Convert (.csv) spreadsheet to MIDI file
converts modified table back to MIDI file.

Prepare Casio Lightning
modifies song that a Casio LK keyboard lights keys for melody and bass

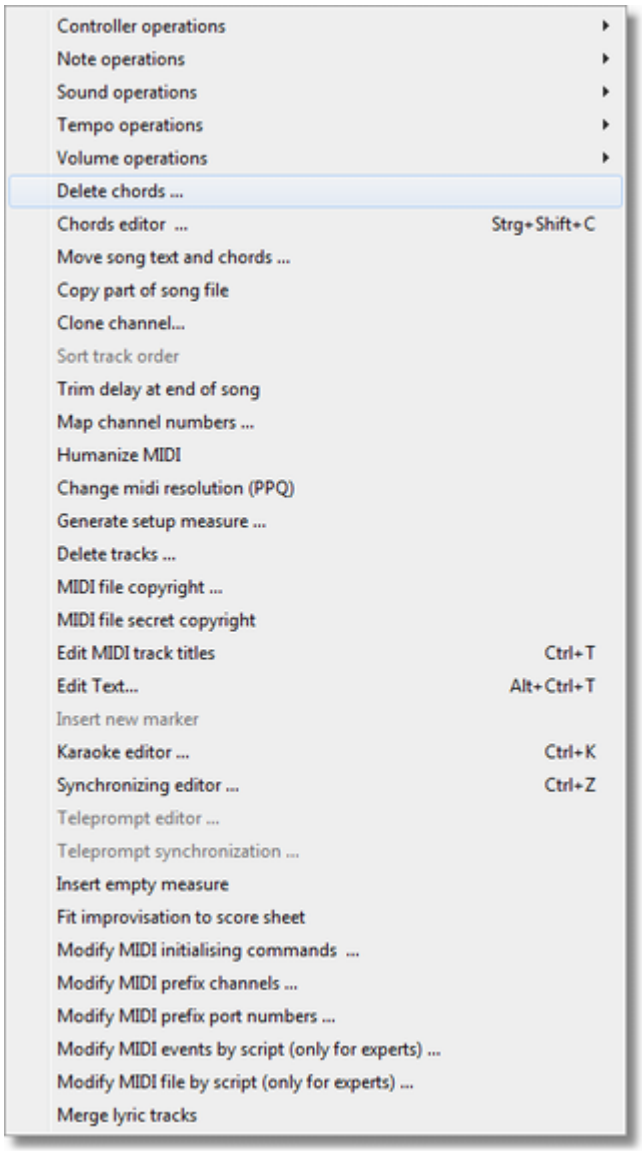
Export song text into .lrc
stores lyrics with time stamps into a text file

Export song lyrics and chords into .crd
stores unsynchronized lyrics and chord names into a text file

Convert MP3 with fixed bitrate (Resample)
converts a MP3 file using constant bitrate.
Microsoft player are using a wrong song duration by estimation which may cause inexactness when jumping to a certain time during playing.
Usually after resampling using a fixed bitrate the problem is solved.
This conversion requires installation of the FFmpeg package.

User operation ...
Operations that are not available in GNMIDI might be programmed for you against a license fee and can be added to this menu.

2.5.4 Modify menu



- Controller operations
- Note operations
- Sound operations
- Tempo operations
- Volume operations

Delete chords

Chords editor ...
modify or insert or delete chords. Using GNMIDI Light license chords editor is not available.

 Copy part of song
Copies or deletes a part from a song (MIDI or MP3)

Clone channel...
Copies all commands of an existing channel to a new channel with optional delay

 Sort track order
change the order of tracks

 Trim delay at end of song
Deletes pauses behind ending of last note

 Map channel numbers ...
renumber the MIDI channels

 Humanize MIDI
add small random timing errors to a MIDI song

 Change MIDI resolution (PPQ)
recalculate MIDI file to a new MIDI resolution

 Generate setup measure ...
add a setup measure in front of MIDI song that initializes parameters

 Delete tracks
remove selected MIDI tracks

 MIDI file copyright
display existing copyright info or add a new copyright notice.

 MIDI file secret copyright
show or add secret copyright information

 Edit MIDI track titles
change track titles

 Edit Text ...
Modify song text and markers and track titles

 Insert new marker
when a MIDI song is playing and currently paused then insert a marker text for this song position.

 Karaoke editor ...
Synchronize song text syllables to melody notes. Using GNMIDI Light license karaoke editor is not available.

 Synchronizing editor ...
Synchronize song text line by line in real time during playing

Teleprompt editor...
changes a teleprompt text (.ptp)

Teleprompt synchronization editor...
Synchronizes some text lines

Insert empty measure
insert an empty measure of given length

Fit improvisation to score sheet
maps free recorded song with help of some markers to bar positions

Modify MIDI initialising commands ...
Modify initialising MIDI commands before notes start ...

Modify MIDI prefix channels ...
Insert, modify, show META prefix channels

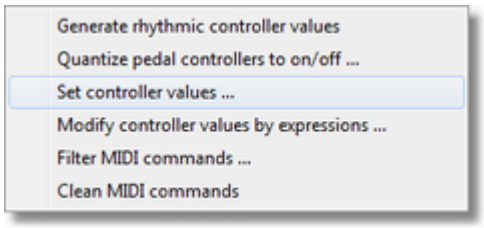
Modify MIDI prefix port numbers ...
Insert, modify, show META port numbers

Modify MIDI events by script (only for experts) ...
use script commands to modify MIDI events with matching criterias

Modify MIDI events by script (only for experts)
use a script program to modify a MIDI song

Merge lyric tracks

2.5.4.1 Controller operations submenu



[in menu Modify/Controller operations]

Generate rhythmic controller values
Add rhythmic controllers that change linear or randomly

Quantize pedal controllers to on/off ...
modify pedal controllers between 0 and 127 to values on or off

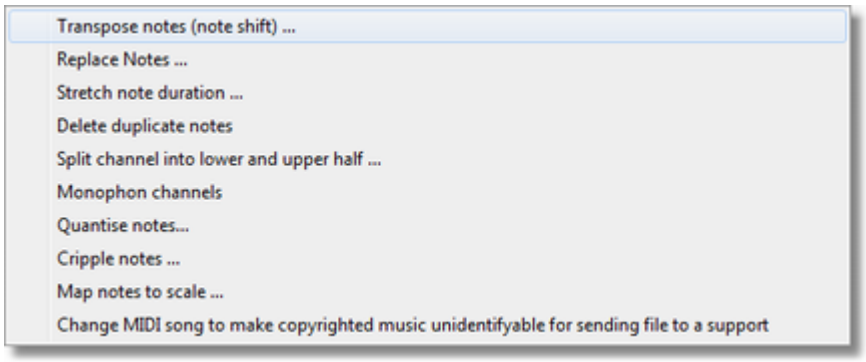
Set controller values
show controller values of selected channels and edit them in dialog

Modify controller values by expressions ...
changes controller values by self defined rules

Filter MIDI commands ...
delete selected MIDI command types

Clean MIDI song
delete and sort some commands

2.5.4.2 Note operations submenu



[in menu Modify/Note operations]

Transpose notes (note shift) ...
transpose notes by some half tones up or down

Replace Notes ...
replace note numbers by other notes

Stretch note duration ...
increase note duration by percentage or incrementing length

Delete duplicate notes
remove duplicate notes that are playing at same time

Split channel into lower and upper half
split notes of a channel into left and right hand

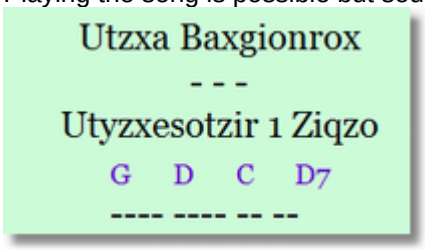
Monophon channels
reduce note polyphony for all channels to single note at a time

Quantise notes ...
align positions and note durations

Cripple notes
encrypt MIDI file that makes it difficult to print score sheet.

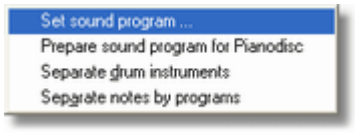
Map notes to scale ...
modify notes that do not match a given scale

Change MIDI song to make copyrighted music unidentifyable for sending file to a support
changes notes and text randomly that the receiver has no use with music self but still can analyse technical problems.
Playing the song is possible but sounds strange.



Hint: unimportant track names are not changed. characters except a-z 0-9 äöüÄÖÜß are not changed

2.5.4.3 Sound operations submenu



[in menu Modify/Sound operations]

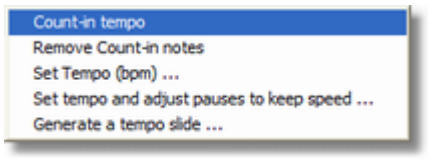
Set sound program ...
assign a new sound for a MIDI channel

Prepare sound program for PianoDisc
convert to MIDI format 0 and set special piano sound for PianoDisc system that the piano keys play magically

Separate drum instruments
move each drum instrument into separate track

Separate notes by programs
move each instrument into a separate track

2.5.4.4 Tempo operations submenu



[in menu Modify/Tempo operations]

Count-in tempo
count-in tempo by adding some drum beats

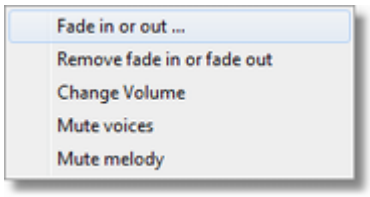
Remove Count-in notes
removes drum notes at beginning of song that are used only for counting in tempo

Set Tempo (bpm) ...
set new constant tempo or change tempo by percentage

Set tempo and adjust pauses to keep speed ...
Set tempo by changing pause times

Generate a tempo slide ...
increase or decrease tempo steadily

2.5.4.5 Volume operations submenu



[in menu Modify/Volume operations]

Fade in or out
Fade a MIDI song at beginning or end

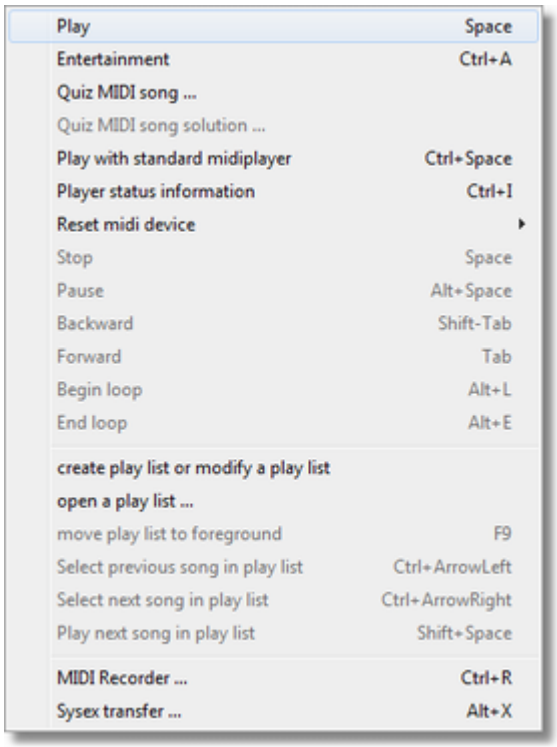
Remove fade in or fade out
finds and removes increasing or decreasing volume changes

Change Volume
change volume or note velocities by percentage

Mute voices
Mute channels or tracks that you want play or sing self

Mute melody
mute melody voice to play it self on keyboard or replace it by a singer

2.5.5 Player menu



 Play
play current MIDI song with internal MIDI player

 Entertainment
play random MIDI or MP3 files for entertainment

Quiz MIDI song
chooses a random song from a MIDI folder and plays a quiz version that plays more and more voices after time.

Quiz MIDI song solution
opens the original MIDI song from previous played quiz MIDI song

 Play with standard MIDI player
play current MIDI song with your favourite MIDI player

 Player status information
get information about playing and navigate the internal MIDI player

 Reset MIDI device
initialize MIDI device by playing an initialization MIDI file

 Stop
stops the internal MIDI player (toggles between play and stop)

Pause
Pause or continue playing current MIDI song

Backward
Continue to play 15 seconds before current MIDI song position

Forward
Continue to play 15 seconds after current MIDI song position

Begin loop sets the beginning position of a loop (when reaching loop end playing continues here)

End loop
sets the ending position of a loop (when crossing the ending position during playing jumps to beginning position.

Create play list or modify a play list
starts play list editing application playlist.exe where you can add songs and store the list in a .lst text file. Using GNMIDI Light license play lists are not available.

Open a play list
Choose a play list file (file extension *.lst). Reads the song list from the text file and opens a document that shows all valid entries. Errors are displayed if file is not found or file format is invalid or songs are not exising. First song is selected in the play list. Using GNMIDI Light license play lists are not available.

Move play list to foreground
if a play list is open then show this play list window in foreground, if it was already in foreground then put the window with currently playing song into foreground

Select previous song in play list

Selects the song before the current selected song. If first song was selected then the last song will be selected and brought into view. The operation is available if current active document is a play list or if there is only a single play list document open. Use key combination ctrl+left arrow.

Select next song in play list
Selects the song after the current selected song. If the last song was selected then the first song will be selected and brought into view. The operation is available if current active document is a play list or if there is only a single play list document open. Use key combination ctrl+right arrow.

Play next song in play list
If the current selected song in the play list is currently playing then the song will be stopped and the next song will be selected and played. Use key combination shift + space. If the current selected song is not playing then this song will be played. The operation is available if current active document is a play list or if there is only a single play list document open.

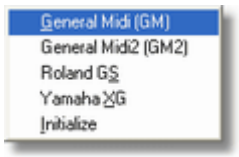
MIDI Recorder ...
record MIDI song via MIDI cable from an external MIDI device

Sysex transfer ...
transfer Sysex (dump) commands through MIDI cable to or from a MIDI device.

Hint: when beginning position is bigger than ending position then both positions are swapped.

Hint: The player status dialog shows the loop position times and has buttons to set begin and end loop

2.5.5.1 Reset midi device submenu



[in menu Player/Reset MIDI device]

General MIDI (GM)
plays GM reset sysex or a user defined reset MIDI file

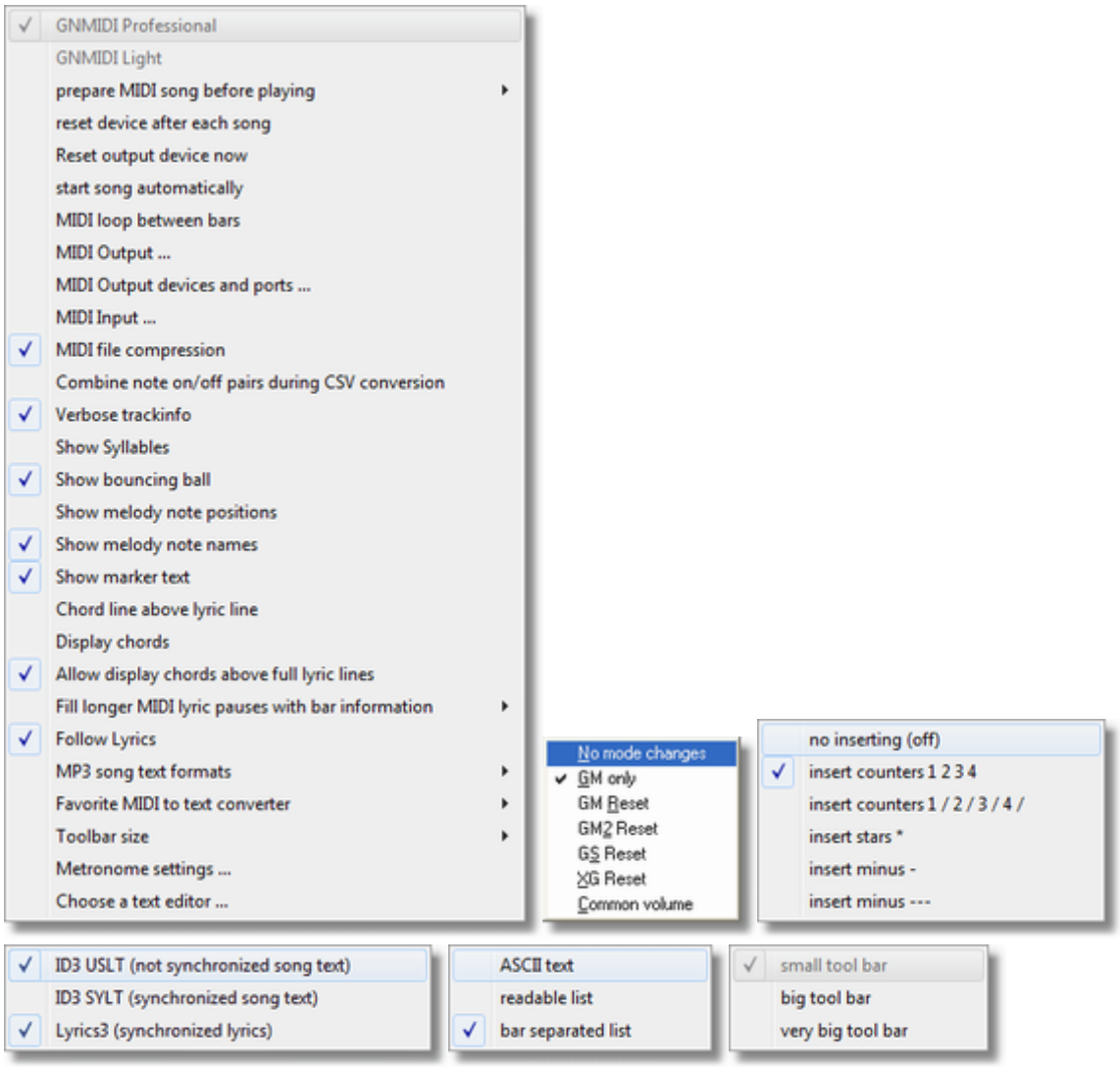
General Midi2 (GM2)
plays GM2 reset sysex or a user defined reset MIDI file

Roland GS
plays GS reset sysex or a user defined reset MIDI file

Yamaha XG
plays XG reset sysex or a user defined reset MIDI file

Initialize
plays initialization MIDI commands or a user defined reset MIDI file

2.5.6 Settings menu



GNMIDI Professional
indicates if Professional license is used. In demo it can be toggled for testing to Light license (e.g. batch operations are then not available).

GNMIDI Light
indicates if Light license is used.

prepare MIDI song before playing
Choose options to prepare your MIDI files before they are played (common volume, device mode initialization ...).

reset device after each song
Choose this option to send SOS MIDI commands whenever MIDI playing stops

Reset output device now
Sends SOS MIDI commands to current output device to stop playing of any notes now .

MIDI Output ...

Select an output device for playing MIDI.

MIDI output devices and ports...
select more output devices for playing parallel to all of them and assign port numbers or port names to them

MIDI Input ...
Select an input device for recording MIDI.

Start song automatically
play song file automatically when opening file or dropping file above application window.

MIDI loop between bars
When setting a new loop begin or loop end position for a playing MIDI song this option causes that instead of the current play time the nearest bar position will be used.
This allows looping between full bars.

MIDI file compression
Use standard MIDI file compression at save operation (suggested).

Combine note on/off pairs during CSV conversion
generates at conversion separate NoteOn and NoteOff lines or a Note line for each pair

Verbose track info
display short or verbose track info

Show Syllables
displays syllable breaks in karaoke text e.g. in-for-ma-ti-on

Show bouncing ball
in karaoke display a ball bounces from syllable to syllable (Warning: this option uses much computer power, do not use it during live performance)

Show melody note positions
positions MIDI melody channel notes on scorelines below lyrics in karaoke display. Choose melody channel(s) using edit description dialog. (Warning: this option uses much computer power, do not use it during live performance)

Show melody note names
melody note names can be displayed for MIDI song when score lines are displayed.

Show marker text
optionally insert marker text lines between karaoke view lines

Chord line above lyric line
text lines with multiple spaces between words are treated as chord lines and are displayed above next lyric line at same time

Display chords
if single chords are available e.g. [Em7] or PSR chords or sysex chords then display them above lyrics

Allow display chords above full lyric lines
full lyric lines have only the start time and no time information during the text line.
Single chords cannot be displayed correctly. Enable this to display these chords inspite of the positioning incorrectness.

Fill longer MIDI lyric pauses with bar information
in karaoke view display extra beats * * * * when lyrics pauses for a full bar

Follow Lyrics
the option must be checked that the lyrics are highlighted during song playing (in karaoke display)

MP3 song text formats

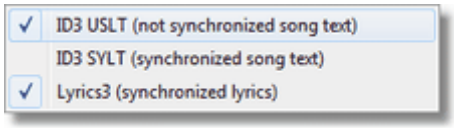
Favorite MIDI to text converter

Toolbar size
choose big or small toolbar buttons (default is using big toolbar buttons for big screen resolution)

Metronome settings ...
choose settings for a metronome type displayed in a metronome toolbar

Choose a text editor

2.5.6.1 MP3 song text formats



(in menu settings)

After synchronizing mp3 lyrics the song text can be stored in different formats (even several).

ID3 USLT (unsynchronized song text)
This format is used by many mp3 players. The text does not contains time stamps.
It is recommended to use this format additionally to a synchronized song text format.
Warning: If selecting this format only then the synchronisation times will be lost.

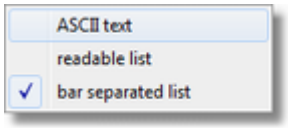
ID3 SYLT (synchronized song text)
This format stores song text with time stamps.

Lyrics3 (synchronized lyrics)
This modern format stores song text with time stamps. It should be used in most cases for storing synchronized lyrics.

By default USLT and SYLT use language code "eng". That could be changed by following .ini setting:

```
[Settings]
mp3EmbedLanguage=ger
```

2.5.6.2 Favorite MIDI to text converter



in menu settings

The  button converts MIDI file to a readable text file and back. It uses the selected favorite text converter in this menu.

The text converters are:
ASCII text a text format that displays exactly the content of the MIDI file with MIDI header, tracks, events in each track.
readable list a list with a header line and a line per each MIDI event. The events are sorted by time (not sorted by track or channel).
bar separated list a list with a header line and bars beginning with === that displays the nom/denom meter of the bar. Notes that continue playing over bars are listed in each of these bars and use the word TIE (indicates that the note continues from previous bar or continues in next bar)

Hint: the A button for converting back to MIDI file detects the used text converter from the text header. Only these valid text formats are supported.

Hint: the text created by these converters can be modified with a text editor (e.g. notepad, notepad2) and the modified file may be converted back to MIDI file (the text format must not contain errors).

Hint: some text converters display the value of an event in different formats (one of these values is more accurate and preferred when both values are given). If you want to modify such a value then delete the alternative value that the conversion result will use only the modified value

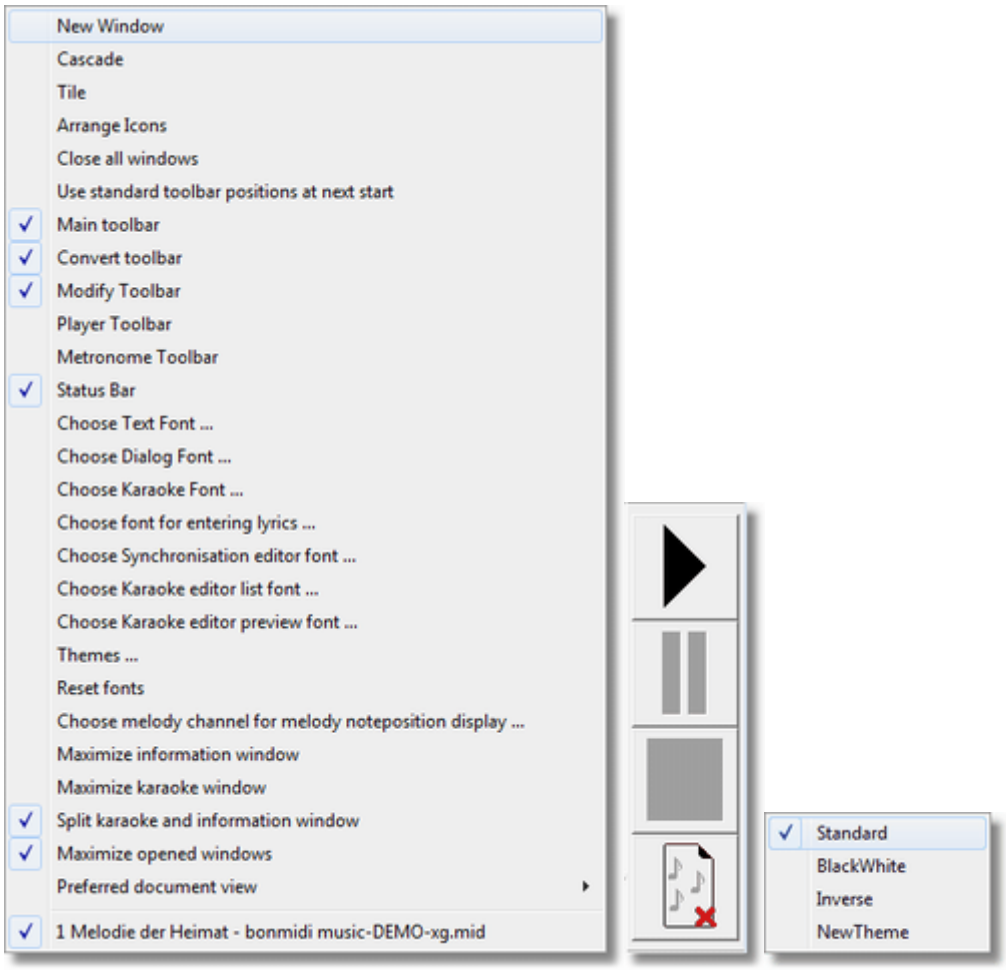
```
TEMPO BPM:116 MSPQ:517240     to change the BPM to 130 remove the MSPQ:517240
value
```

(the MSPQ microseconds per quarternote is stored in MIDI file,for changing BPM beats per minute is often used)

```
Note    48      258ms   T1      C10     11units 59ms    g#3     Vel96   Offvel64
```

(the note position are given in MIDI units and milliseconds 48 and 258ms, the note duration is given in MIDI units and milliseconds 11units 59ms)

2.5.7 Window menu



Use standard toolbar positions at next start
do not store toolbar positions at end of program (once only).
All toolbars get default dock positions at next program start.

Main toolbar
Show or hide the main toolbar

Convert toolbar
Show or hide the convert toolbar

Modify toolbar
Show or hide the modify toolbar

Player Toolbar
Hides or shows player dockable toolbar

Metronome Toolbar
Hides or shows metronome dockable toolbar

New window

Opens another window with same content

Cascade
Windows are moved so that one overlaps the other.

Tile
Windows are moved one beside the other

Arrange Icons
Minimized windows are arranged to get better overview

Close all windows
Quickly close all windows without saving changes

Status Bar
Hides or shows status bar at bottom of application

Choose Text Font ...
text font used in MIDI information window

Choose Dialog Font ...
text font used in dialog content.
Default font is MS Sans Serif 8.
Do not choose too big font because many dialog items have fixed size and the item text will then be cut.
A nice font is e.g. Comic Sans MS 9

Choose Karaoke Font ...
text font used in karaoke display

Hint: You could also change font using Themes

Choose font for entering lyrics
text font used at beginning of Karaoke editor and Synchronisiation editor to enter or modify lyrics. The font should have fixed pitch (all characters have same width) that the chord line feature in the synchronisation editor can be used.

Choose Synchronisation editor font ...
select a font and font size for displaying lyric lines in synchronisation editor dialog

Choose Karaoke editor list font ...
select a font and font size for displaying lyric syllables in karaoke editor dialog

Choose Karaoke editor preview font ...
select a font and font size for displaying preview lyrics in karaoke editor dialog

Reset fonts
set dialog fonts back to initial fonts

Themes ...
shows the collected list of themes found in *.theme text files (in program folder, in appdata folder, in documents folder). A theme is a collection of display settings of the karaoke view (e.g. font, background color, text colors, karaoke ball color, karaoke ball size). Some standard themes are available in program folder (file standard.theme). They can be copied to user themes and then edited with the theme editor.

Choose melody channel for melody noteposition display
find a melody channel for current song that should be used in karaoke display for note lines

Maximize information window
splits the current window so that only the upper information part is visible

Maximize karaoke window
splits the current window so that only the bottom karaoke part is visible

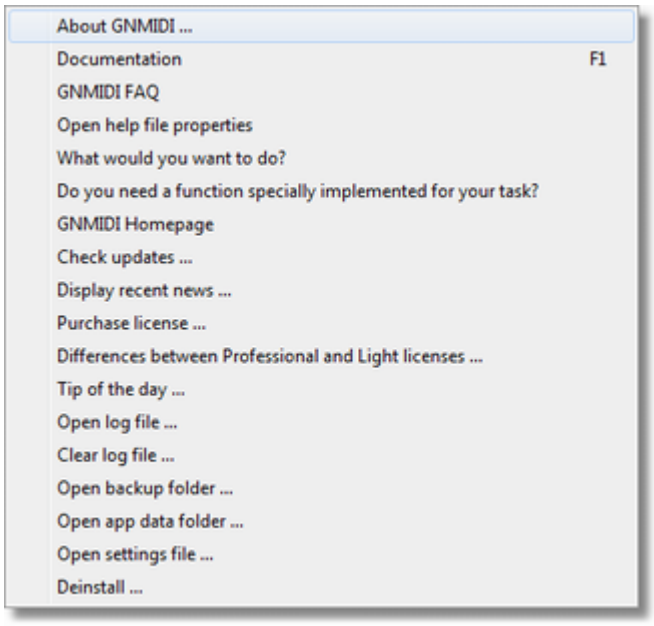
Split karaoke and information window
splits the window into half so that both karaoke and information parts are visible

Preferred document view
choose a view INFO,INFO|KAR,KAR that is preferred if the standard behaviour is not wanted

Maximize opened windows
normally new windows are opened smaller and positioned near the previous opened window. With this option a new document will be automatically opened maximized. The karaoke text is better readable and can be displayed with bigger font.

Window list
Shows list of currently open document windows, the active document is checked. Selecting a document window will bring it into front and activate it.

2.5.8 Help menu



About GNMIDI ...
displays copyright and version number

Documentation
open help document for current topic (key F1 can be used)

GNMIDI FAQ
open web page that contains answers to frequently used questions to GNMIDI

Open help file properties
If the program help (opened with F1) does not show text the reason is a strange security behaviour of Microsoft Windows that also blocks help file text files (containing only text, pictures, links) that were downloaded together with the program (usually .zip). In the file properties page General will contain a security warning and with button "Unblock" the needless blocking can be unblocked and then the help text should show the missing text.

What would you want to do?
offers MIDI help

Do you need a function specially implemented for your task?
opens a web page from gnmidi.com that explains user functions in GNMIDI

GNMIDI Homepage
goes directly to https://www.gnmidi.com/gnmidien.htm (here you can download newer program versions)

Check updates
opens a web page at https://www.gnmidi.com that checks if there is an update version available. For registered users it also checks if there is a newer update that the purchased license does not include.

Display recent news ...
opens a web page in your browser with the latest news about GNMIDI

Purchase license ...
order a single user license for using GNMIDI

Differences between Professional and Light licenses ...
help about licenses with comparism of differences between Professional and Light licenses

Tip of the day ...
read some useful hints about using this software

Open log file ...
GNMIDI writes a log file with information which commands were used and errors and warnings if something happened.
This file can be viewed in a text editor using this operation.

Clear log file ...
deletes the content of the gnmidi3.log text file

Open backup folder ...
When saving modifications using File/Save operations backups of the original files are kept in backup folder.
It opens the backup folder with file explorer. You may delete unnecessary backup files or restore a previous version by renaming and moving a file.

Open app data folder ...
opens the GNMIDI application data folder (contains log file, license file, demo songs, ...) with Windows file explorer

Open settings file ...
open the GNMIDI settings file (gnmidi.ini) with notepad editor

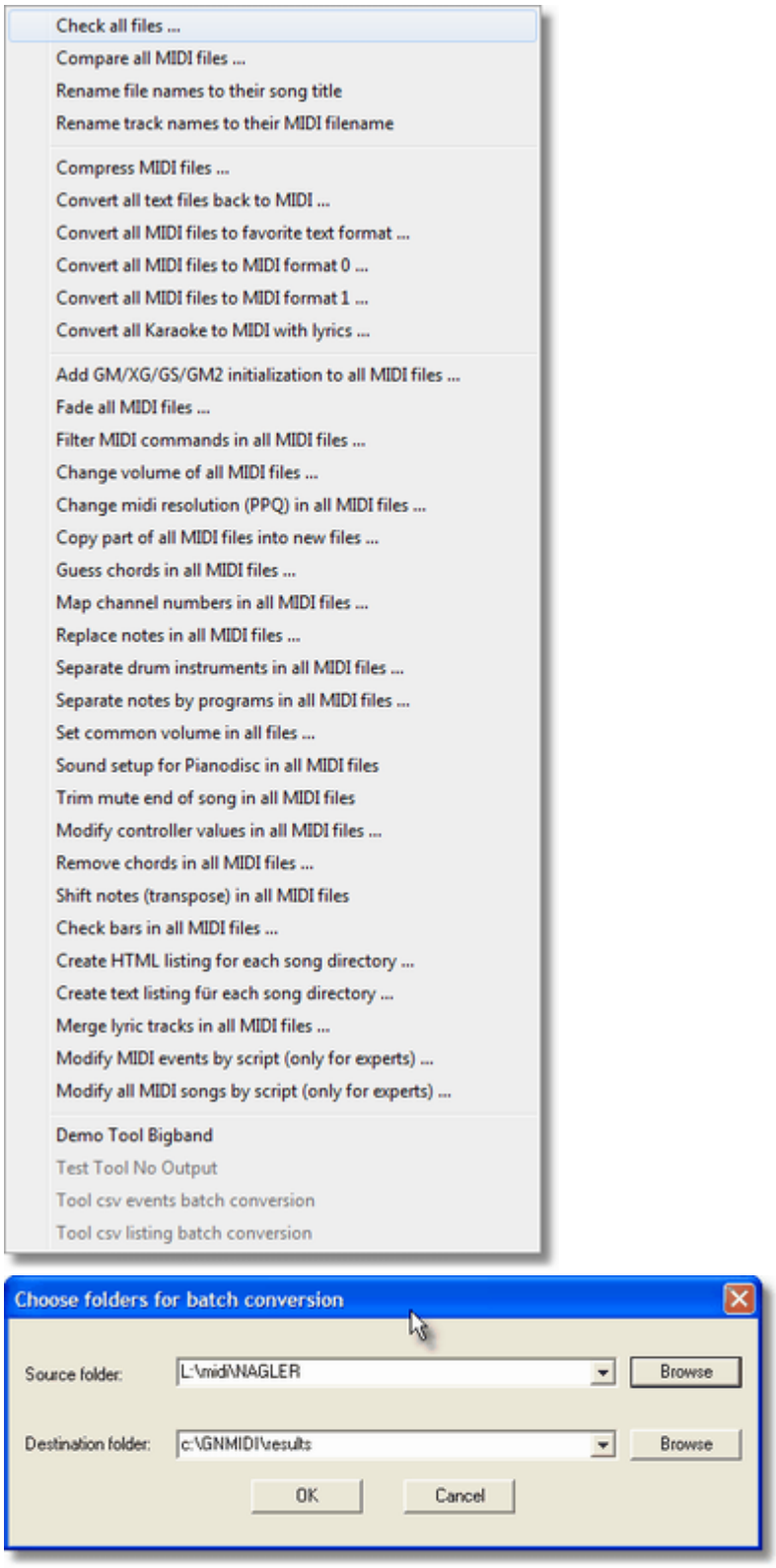
Request access code for this computer
When you have installed a full license this option might appear to register the current computer for use with this license.
This counts the number of computers used with your license file (max. 5 computers for a license).
Requesting an access code can be done online (webpage) or offline (by email).
GNMIDI might ask self for entering the request code some weeks after your order.

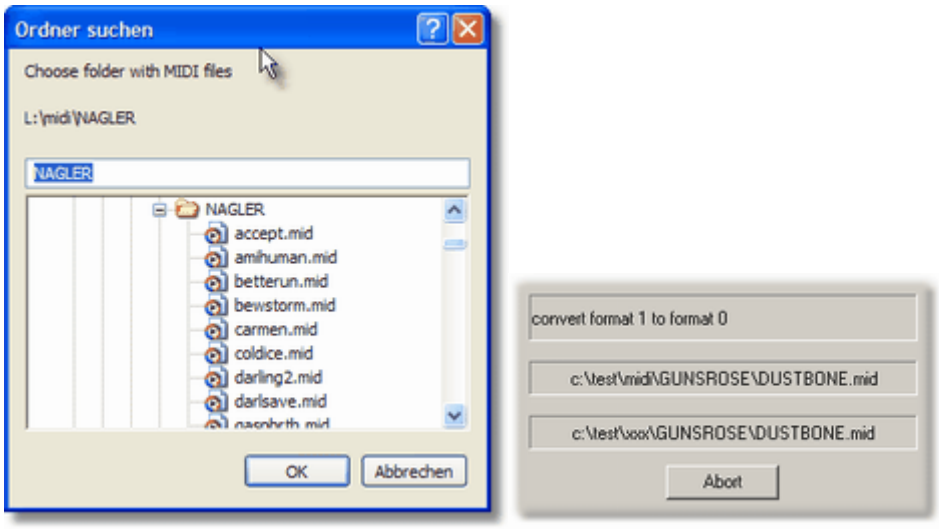
Deinstall
Opens Windows system deinstall programs app where GNMIDI can be deinstalled when MSI installation was used.
For zip installation delete the files in the folder where you unzipped GNMIDI. Some more files are in application data folder that you can open with help menu entry.

3 GNMIDI operations

3.1 Batch operations for favourite operations


[in menu File]


Check all files ...
Compare all MIDI files ...
Rename file names to their song title
Rename track names to their MIDI filename
Compress MIDI files ...
Convert all text files back to MIDI ...
Convert all MIDI files to favorite text format ...
Convert all MIDI files to MIDI format 0 ...
Convert all MIDI files to MIDI format 1 ...
Convert all Karaoke to MIDI with lyrics ...
Add GM/XG/GS/GM2 initialization to all MIDI files ...
Fade all MIDI files ...
Filter MIDI commands in all MIDI files ...
Change volume of all MIDI files ...
Change midi resolution (PPQ) in all MIDI files ...
Copy part of all MIDI files into new files ...
Guess chords in all MIDI files ...
Map channel numbers in all MIDI files ...
Replace notes in all MIDI files ...
Separate drum instruments in all MIDI files ...
Separate notes by programs in all MIDI files ...
Set common volume in all files ...
Sound setup for Pianodisc in all MIDI files
Trim mute end of song in all MIDI files
Modify controller values in all MIDI files ...
Remove chords in all MIDI files ...
Shift notes (transpose) in all MIDI files
Check bars in all MIDI files ...
Create HTML listing for each song directory ...
Create text listing für each song directory ...
Merge lyric tracks in all MIDI files ...
Modify MIDI events by script (only for experts) ...
Modify all MIDI songs by script (only for experts) ...
Demo Tool Bigband
Test Tool No Output
Tool csv events batch conversion
Tool csv listing batch conversion
Choose folders for batch conversion
Source folder:
L:\midi\NAGLER
Browse
Destination folder:
c:\GNMIDI\results
Browse
OK
Cancel



[in menu File]

A batch operation is an operation that is applied to all MIDI files in a MIDI folder (including all sub folders).
Using GNMIDI Light license batch operations are not available.

Operations usable for batch execution
Menu File contains item Batch conversion that displays the menu of currently available operations. When no MIDI document window is open then the menu and the toolbar displays only batch operations.

Steps

1. a dialog to fill required **parameters for the operation** appears and must be filled and completed with OK (this does not occur if the operation has no parameters). It is the same dialog that appears when performing the operation on a single MIDI document.
2. a **source folder** must be chosen that contains the original files that should be converted. Select folder from previous operations using drop down combo box.
3. a **destination folder** must be chosen (the folder must exist and should be empty), use *create new folder* to create a new folder (some operations don't need a destination folder when they don't generate output files e.g. Check all MIDI files). Select folder from previous operations using drop down combo box.
4. You will be asked if the conversion should be started, **continue with Yes** or **cancel** the operation with No. You should check if the displayed path information is correct.
5. **batch execution starts** after the above steps are accepted with OK or yes buttons.
6. The batch conversion displays the operation and current source and destination file. The batch conversion can be **aborted** when needed.
7. At completion of the batch conversion a text file will be displayed (with Notepad editor) with important information about certain file conversions (e.g. errors, warnings, information). You should take care about the information, and you can save or print the text file with Notepad editor.
8. The **destination folder contains the resulting files** with same names at successful conversion (some operations might change file extension automatically e.g. MIDI -> Text renames to *.txt).

Hint: The source MIDI folder must be different to the destination MIDI folder, this operation writes the resulting files into a new folder instead of overwriting your files. **Never try to convert from a folder into same folder, this could destroy your MIDI files!**

Hint: Choose a destination drive that has enough free disk space, else the operation might not complete successfully.

Demo limitation
the unregistered program demo (for testing only) limits the number of conversion for a batch operation. The registered program has no conversion limitation.

Functions
Check all MIDI files
Compare all MIDI files
Rename file names to their song title
Rename track names to their MIDI filename

Compress MIDI files
Convert all ASCII text files back to MIDI
Convert all MIDI files to favorite text format
Convert all MIDI files to MIDI format 0
Convert all MIDI files to MIDI format 1
Convert all Karaoke to MIDI with lyrics

Add GM/XG/GS/GM2 initialization to all MIDI files
Fade all MIDI files
Filter MIDI commands in all MIDI files
Change volume of all MIDI files
Change midi resolution (PPQ) in all MIDI files
Copy part of all MIDI files into new files
Guess chords in all MIDI files
Map channel numbers in all MIDI files
Modify controller values in all MIDI files
Replace notes in all MIDI files
Remove chords in all MIDI files ...
Separate drum instruments in all MIDI files
Separate notes by programs in all MIDI files
Set common volume in all MIDI files
Sound setup for Pianodisc in all MIDI files
Trim mute end of song in all MIDI files
Shift notes (transpose) in all MIDI files
Check bars in all MIDI files ...
Create HTML listing for each song directory ...
Create text listing für each song directory ...
Merge lyric tracks in all MIDI files ...
Modify MIDI events by script (only for experts) ...
Modify all MIDI songs by script (only for experts)

User operation ...
Operations that are not available in GNMIDI might be programmed for you against a license fee and can be added to this menu.

3.2 Open a MIDI file


[in menu File]

A file dialog starts and you can choose the folder location and name of the MIDI file. Use the directory list and filename list to find it. Once you have specified an existing file, the MIDI information will automatically be analyzed and shown in a new window with filename as title.

If the MIDI song contains karaoke text then the information window has two parts:

- upper part shows MIDI information
- lower part shows karaoke song text.

Both are separated by a **splitter**, which you can move up or down with left mouse button to get more space for upper or lower part.

splitter

It is also possible to open a text file that contains readable MIDI text (generated by MIDI to ASCII converter).
This text will be shown in the window. If the text is very large then it is cut at end, use Ctrl-E to display the full text with notepad editor or open the text with a text editor that can handle huge text files.

A vertical scroll bar on right side of window is available if the information in the windows is larger than the display size. You can scroll the text up and down with mouse and on some systems with a wheel mouse. A horizontal scroll bar on bottom of the window is available if the information width is larger than the
window.

If the file is not a MIDI file or if the MIDI file is corrupt then an error message is shown instead of the MIDI file information. Try to repair the MIDI file.

The file menu contains also the list of **recent loaded files**, this can be used to reopen one of these files quickly.

Importing other file formats

.kar
Karaoke files (.kar) are valid standard MIDI files, so just open them as MIDI files.

.rmi
RIFF MIDI files are multimedia archives, which can contain one or more MIDI songs. Using Open MIDI file will only import first standard MIDI song.

.st3
Karaoke files in format .st3 (only version 4.5 files supported) can be imported to MIDI song with song text.

.txt
Text files converted by function MIDI to ASCII text can be displayed and then converted back into MIDI file. Big files will not be displayed completely, use a text editor that can display and edit big text files instead.

.rtttl
small text files that contain ringing tones in format RTTTL will be displayed in a window. Such a file can be converted using function RTTTL to MIDI if content is valid RTTTL format.

.mp3
ID3 tag Information of a MP3 music file will be displayed in information window. Available MP3 song text is displayed in karaoke window.
A few operations can be applied to MP3 music files with GNMIDI (e.g. play, entertain, synchronize lyrics, find text).
.mp3 lyrics are not loaded if

- ID3v2 block is missing
- ID3v2 data is not correctly formatted according to ID3v2 specification
- lyrics data is encrypted (rarely used)
- lyrics data is compressed (rarely used)

3.3 Close a MIDI file

[in menu File]

You can close a MIDI document window at any time, even if the song is currently playing. When closing new generated or modified MIDI files (marked with a * in title) then the program asks if you want to save it

Yes
will ask for a filename

No
does not save and removes the MIDI

Cancel
the window will remain open

If you want to close all MIDI files without saving modified ones then use Close all windows in window menu. Save your important modified files before exiting the software!

Each document window has a close button on right top of its caption that can also used for closing the MIDI document window.

3.4 Play a MIDI file


[in menu Player]

Play command will start the internal MIDI player for the MIDI file in window that currently has the focus (this window has a highlighted caption). It will automatically stop playing the file when closing the file or starting a new play command.

Space key starts playing the song.

With menu settings you can turn on options that the MIDI song will be prepared for your device before playing.
With menu settings you can also select the output device that should play this song.

Player status window displays information about playing and contains buttons to navigate.

Hint: since version 2.48 GNMIDI uses an own internal MIDI player (the one that is also used by GNMixer) instead of the formerly used Microsoft MCI player to play MIDI files. If you want to use again the MCI player (for whatever reason) you can set this back with following gnmidi.ini setting:

```
[Settings]
PlayerUseMCI=1
```

Hint: since version 2.48 GNMIDI can also play MP3 songs (through Microsoft MCI) and display synchronized song text.

Status bar (bottom) shows sometimes an information at left side and play time and song duration time

and for MIDI songs bar position field when the song is playing or paused:

Hint: click with left mouse button into the status bar field time position or bar position will show an edit field instead. Enter a text or bar position and press enter key to go to continue playing at this MIDI position. Use ESC key to close the edit box if no jumping is wanted. The edit field accepts following position formats:

```
h:mm:sec.millisec
mm:sec.millisec
mm:sec
sec.millisec
millisec ms    e.g. 12350ms
unit    e.g. 0 is song begin
measurenr.beatnr.unit  e.g. 1.1.0 is song begin
beatnr.unit  e.g. 2.0 is second beat in first measure
```

During **playing MP3 songs** the Microsoft player works internally with often **wrong estimated song duration** (instead of calculating the correct song duration). GNMIDI tries to solve the inexactness but sometimes the estimated song duration is so far and wrong away from correct song duration that the inexactness can be heard when jumping to a certain real time and the played music does not play what is really what should be heard that time.
Using the MP3 operation convert MP3 with constant bitrate (Resample) in menu convert an alternative MP3 file can be created that usually does not cause the inexactness at guessing the song duration.

3.5 Stop the MIDI song player

[in menu Player]

The stop command stops the MIDI song currently played by the internal MIDI player.
It does not stop an external MIDI player (for this use the stop command on your external player).

Space key stops current song playing.

3.6 Save a MIDI file

[in menu File]

Save
The save command must be used to store a new generated or modified MIDI file.
During editing modified files are kept in a temporary files folder and will be deleted when closing the application if they were not saved.

Save As ...
The save as command is used to store the file with a new name or to an other folder. It will ask for choosing the new file name and location.

Hint:
Working folder changes after Save As operation when a new location is selected. For copying files to a slow drive (e.g. floppy disk) better use Windows Explorer than Save As because future operations might access the slow drive and that could slow down the work speed unnecessarily.

3.7 Change MIDI format

[in menu Convert]

This operation converts MIDI format (also called version or type) from 0 to 1 or 1 to 0.

Format 0:
all MIDI commands are in a single track and sorted by time. Some devices only support this file format.

Format 1:
MIDI commands are assigned to more tracks, each track should have not more than one channel. First track is reserved to conductor.

Convert all MIDI files to MIDI format 0:
These conversions are also available as batch conversion. Using GNMIDI Light license batch operations are not available.

The MIDI information window tells the file format of the MIDI file.

3.8 Check and repair a MIDI file

[in menu Analyse]

The operation checks the MIDI file against the Standard MIDI file format. For a valid MIDI file it will display the message

```
MIDI file has valid standard MIDI format. Repair not needed.
```

When opening a corrupt MIDI file an error message will be displayed in the MIDI information window. The operation can repair common MIDI file problems (e.g. invalid command parameters, truncated MIDI file), but it can not repair severe errors (e.g. if lost data at beginning of file) and repairing might lose data.

The operation is also available as batch operation. Using GNMIDI Light license batch operations are not available.

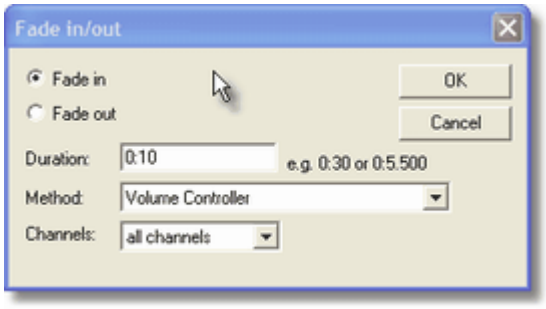
A **Mp3 file** can also be verified if FFmpeg package is installed.
It tells result "dangerous" if the *.mp3 file contains an executable program instead of music.
Hint: repairing is not possible with mp3 file.

3.9 Convert RIFF MIDI (.rmi) to standard MIDI file (.mid)

[in menu Convert]

This operation is only available when opening a file with extension .rmi (RIFF MIDI file). Those files are multimedia archives that can contain MIDI documents and other multimedia documents like WAV, AVI. The conversion extracts a contained MIDI file from the archive.

3.10 Fade a MIDI song



[in menu Modify/Volume operations as fade in or out]

Fade operation increases or decreases volume level at beginning or ending of a MIDI song steadily so that the loudness fades linear.

The fade dialog occurs and you can choose if you want to fade in (at beginning of song) or fade out (at end of song). A duration of 10 seconds will be offered and can be changed. By default the volume changes by inserting increasing/decreasing volume controller (#7), you can choose between 3 different methods that affect song volume. Usually all channels are faded, choose a channel to operate only on this channel.

Fade in:
fades from beginning of song for the specified duration (from 0% to 100%, increasing note velocities).

Fade out:
fades till end of song for the specified duration (from 100% down to 0%, decreasing note velocities).

In **Duration** field you must enter a valid time in one of following formats:
- `minutes:seconds.milliseconds (e.g. 1:25.300)`
- `minutes:seconds (e.g. 0:10)`
- `seconds.milliseconds (e.g. 5.500)`
- `seconds (e.g. 10)`

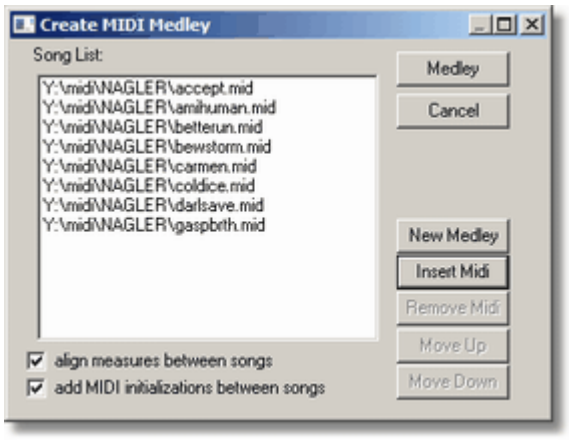
Method
you can choose one of following kinds of changing volume , we suggest modifying volume controller
- `Volume Controller:  MIDI controllers #7 are inserted`
- `Expression Controller:  MIDI controllers #11 are inserted`
- `Note Velocities:  Note velocities of each note are adjusted`

Hint:
Modifying note velocities can influence voice of instruments on some devices.

Channels:
By default all channels are faded, optionally you can choose a certain channel to fade.

Fade in/out can be used with mp3 files if FFmpeg package is installed.
Hint: option channels is ignored for mp3 file

3.11 Make a MIDI medley



[in menu Convert]

The MIDI medley operation displays a dialog where you can enter a list of MIDI files. After the list is ready and sorted in your preferred order press the **Medley** Button. Then the operation checks MIDI format of all input files and converts all MIDI files temporarily to format 0. If there are invalid files or files that are not format 0 or format 1in the list then the operation stops and the first invalid file is opened. You could try to repair the file. All MIDI files are concatenated to a file containing the songs in a sequence (medley).

No pauses are removed or inserted. If you want a pause between the songs then insert a small MIDI song that contains only the pause.
The MIDI medley will be opened as a new document.

The list box remembers recent used filenames, use **New Medley** button to clear this list when starting a new medley.

Use **Insert MIDI** to add one or more files to the song list, the new files will be inserted before selected entry in the list.

Use **Remove MIDI** to delete the selected filename from the song list.

Use Move Up and **Move Down** to sort the list entries.

Align measures between songs
this Option can be selected to insert pause MIDI units that the first bar of next song is correctly aligned. Otherwise there might be a smaller bar then expected (e.g. 3/4 instead of 4/4).

Add initializations between songs
When the option is checked (default) then controllers and other important MIDI parameters are added to initialize settings of next song to avoid bad sound if the author of an input file has not

initialized a parameter and in medley value of recent song is used for the next song. This option can be turned off if you are sure that all necessary MIDI parameters are initialized for each song and don't conflict. Especially when the result is not used as medley but as one song combined from several parts, then the extra initializations are not necessary (could cause a small delay).

3.12 Play MIDI with favorite MIDI player

[in menu Player]

This operation starts the application that is associated with the file extension of current MIDI file (usually .mid, could also be .kar). It automatically adds the full location and name of current MIDI document as first parameter.

Applications can be associated to file extensions with help of Windows Explorer.

The external player can be started with Ctrl+Space key.

3.13 Split MIDI medley

GNMIDI combines two or more songs into a medley with generate MIDI Medley operation.
It adds markers that indicate where the original parts begin.
For a medley some operations are more difficult than doing operations with the parts self.

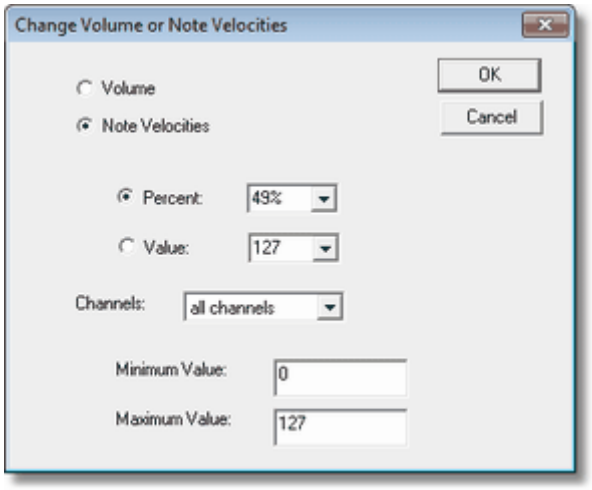
This operation splits a medley at given markers so that they can be combined after part modifications again.

The result file names will be:

`origfilename###.mid` where ### is a number 001, 002, ...

If input midi folder is writable then the files will be in this folder else the files will be in Windows temporary folder.
After successful splitting the operation writes a message with the generated full file names and opens the folder where the results are.

3.14 Change Volume (or Note velocities)



[in menu Modify/Volume operations]

You can choose between modifying volume by **volume** controllers or **note velocities**. You can change original values percentually or set them to a new constant value.

Hint: Prefer modifying volume controllers but if the volume controllers are already at maximum level then only way might be to increase note velocities.

Percentage
Enter a percentage between 0% and 1000%. Percentage 100% does not change volume.

Value
Enter a value between 0 and 127. Value 0 is not allowed for setting note velocities, this value has a reserved meaning in MIDI.

Channels
Optionally choose a channel between 1 and 16 or all channels (default is all channels).

Minimum Value
Values smaller than minimum value will be set to minimum value (default: 0)

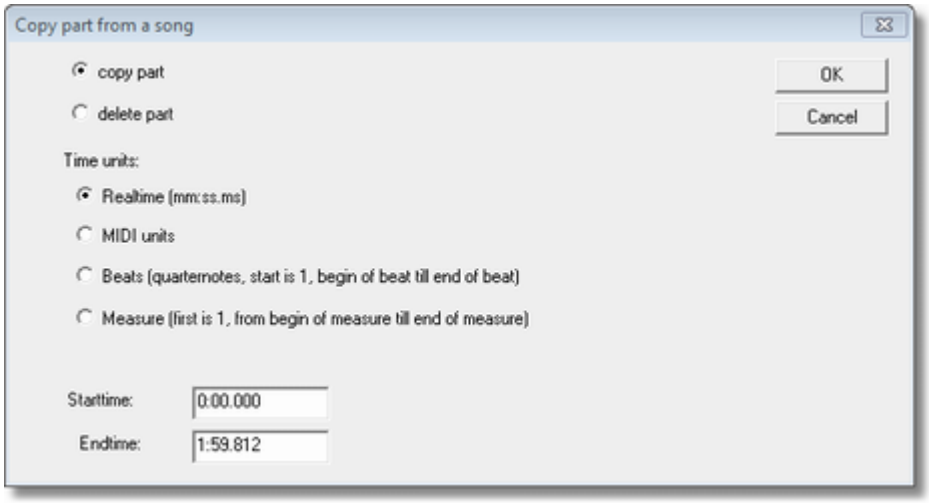
Maximum value
Values higher than maximum value will be set to maximum value (default: 127)

Turn off "adjust volume to common level" option in menu settings, when you test the volume changes with GNMIDI player, since this option will automatically adjust volume of played song to a common level.

Volume can be changed also for **mp3 files** if FFmpeg package is installed.
For mp3 file only relative volume changes can be done e.g. 120% or +3dB.
Use option Value 103 for +3dB and 97 for -3dB (percent or value 100 means no volume change for mp3).
Too high or too low values might cause error during FFmpeg conversion.

Hint: options Note velocities, minimum value, maximum value, channels are ignored for mp3 file

3.15 Copy part from a song



[in menu Modify]

Copy part

This operation copies a part of MIDI or MP3 song into a new MIDI file.
The range start-end will be copied into a new MIDI file. Notes that play at start or end position will be cut to a smaller note inside the new range. Settings are not removed, so the part will play with original settings. Duplicate controller settings before the start position are combined. Pitchbend commands before start are combined.
Hint: copy time part of mp3 song requires installed FFmpeg package for Windows.
Hint: for MP3 only realtime range can be used

Remove part

This operation deletes a part of the MIDI song.
Notes and lyrics that are fully inside middle part are deleted, notes overlapping middle range are shortened, pauses in middle part are deleted. Controllers, Pitchbend inside the middle part are kept and optimized so that the settings necessary for playing right part correctly are not lost.
Hint: copy time part of mp3 song requires installed FFmpeg package for Windows.
Hint: for MP3 only realtime range can be used

Time units:

Choose one of the 43 position formats for specifying start and ending of the cut range. Usually real time is wanted.

Realtime (mm:ss.ms)

Time position relative to the beginning of the MIDI song.
You can specify (minutes:seconds:milliseconds (e.g. 3:26.975) or minutes:seconds (e.g. 3:27) or seconds (e.g. 207).

MIDI units

Precise positions in MIDI file units (a number between 0 and last unit in MIDI song). MIDI units are smallest usable note length or pause length in a MIDI song (e.g. at song resolution 96 a MIDI unit is a 1/384 note length).

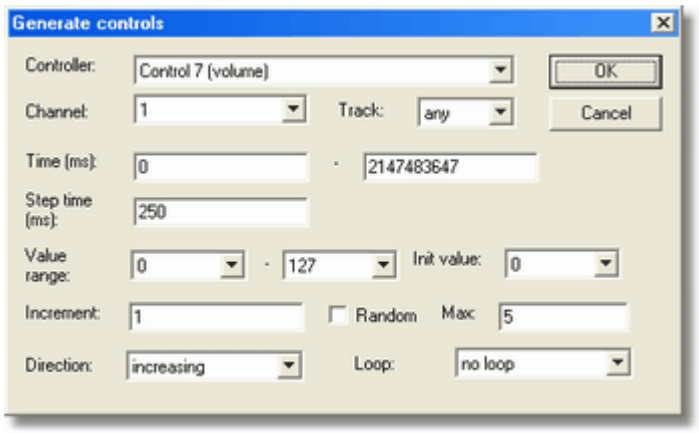
Beats (quarter notes)
Positions are specified in beat numbers (a number between 0 and last beat in MIDI song) since start of song (a beat is a quarter note).
The range includes the positions from begin of first specified beat number till end of second specified beat number.
E.g. 1-1 is begin of beat 1 till end of beat 1

Measures
Positions are measure (=bar) numbers (a number between 1 and last measure number) since begin of song.
The range includes the positions from begin of first specified measure number till end of second specified measure number.
E.g. 1-1 is begin of measure 1 till end of measure 1

Starttime:
Endtime:
Enter start and end position in selected position format.

Hint:
Deleting some part from the beginning of song will remove all pauses of this part and this can influence the behavior of reset sysex commands badly. Initialisation commands running parallel to such reset sysex commands might finish earlier than the reset sysex finish and this could cause that those initialisations are not done during playing. It can be often noticed when some sound program is used but plays default piano sound.

3.16 Generate control events in rhythmic or random way



[in menu Modify/Controller operations]

This operation adds controllers that behave random or according to a linear rhythmic way. The parameters specify which controller should be added and how the controller values must behave during time
e.g.

- increase volume from one position to another
- change balance steadily from left to right and back
- modify expression controller

Controller:

Choose a controller number from the list. Controller commands of this kind will be generated. Usually volume or balance controllers are preferred here.

Channel:
choose one channel from the list, the controller commands will be added to this channel.

Track:
track number can be selected optionally (by default all tracks are considered). It could be useful if the chosen channel number is used by more than one track.

Time:
enter two millisecond values to generate controllers only within a certain time range (default is full MIDI song time range). Second value must be greater than first one (e.g. 30000-60000 is range between second 30 and 60).

Step time (ms):
specifies the delay between two generated controllers (two steps) in milliseconds. When the value is small then huge count of controllers might be generated.

Value range:
Controller values must be in range 0-127. Optionally you can force the generated controller values to be in an other value range (e.g. 64 - 80).

Init value:
Specify the initial controller value that should be generated at start.

Increment:
Specify an incremental value if the controller values should be increased or decreased by a value.

Random:
Check that option if the values should be generated by random instead of linear incrementation.

Max:
Random generated values maximum difference to previous generated value. This value restricts the random distance to the next generated value.

Direction:
- increasing next value is calculated by adding the increment value
- decreasing next value is calculated by subtracting the increment value
- random next value is calculated by adding or subtracting the increment value

Loop:
- no loop stop generating values when the next generated value would exceed the value range
- loop cycle when value exceeds value range then restart with initial value again
- loop up and down direction when value exceeds value range in one direction then continue in reverse

3.17 Show short or verbose track information

[in menu Settings]

With this option you can choose between short or large information about MIDI tracks that will be displayed in each MIDI information window.

short track information

A short track information line contains
track number [channel number]: "track title" (initial program)

```
Track 1 [no channel]: "Liebesspieler (Die Toten Hosen)"
Track 2 [no channel]: "Soft Karaoke"
Track 3 [no channel]: "Words"
Track 4 [1]: "Melody" (Acordion)
Track 5 [3]: "whistling" (Whistle)
Track 6 [4]: "horse running" (WoodBlok)
Track 7 [5]: "choir" (AahChoir)
Track 8 [6]: "solo guitar" (Distortd)
Track 9 [10]: "Drums" (GM Drums)
Track 10 [12]: "Bass" (FngrBass)
Track 11 [13]: "Western guitar" (SteelGtr)
Track 12 [14]: "Western Guitar 2" (NylonGtr)
Track 13 [15]: "Mute guitar" (MuteGtr)
```

Example:

```
Track 4 [1]: "Melody" (Acordion)
```

that means: track number is 4, contains only one channel with number 1, track title is "Melody" and initial program (sound) is Accordion.

verbose track information

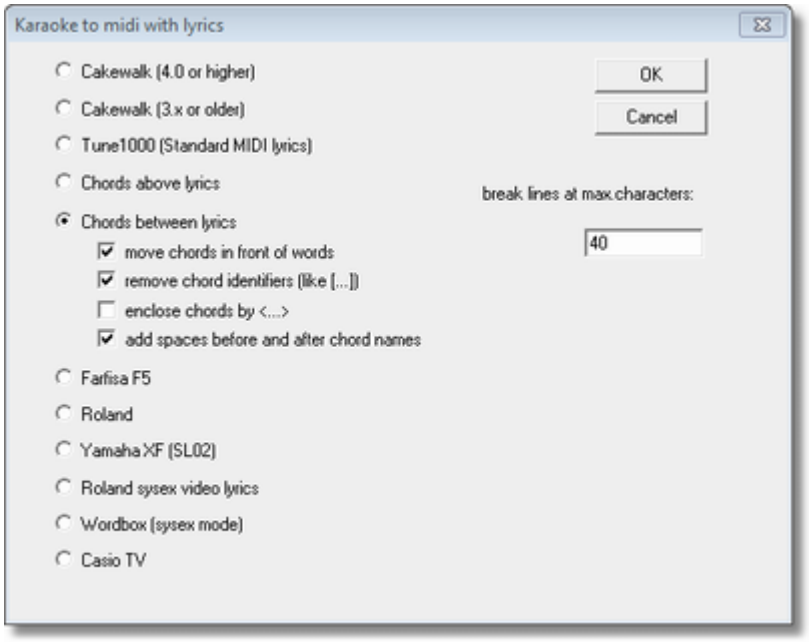
A verbose track information contains
track number:
- channel: channel number
- Name: "track title"
- Program: sound program
- Volume: volume (0-127)
- Balance: balance (0=left, 64=middle, 127=right)
- chorus: chorus effect level (0-127)
- reverb: reverb effect level (0-127)
- notes: deepest and highest note used (note range size in half tones)

```
Track 1:
  channel: no channel
  Name: "Liebesspieler (Die Toten Hosen)"
Track 2:
  channel: no channel
  Name: "Soft Karaoke"
Track 3:
  channel: no channel
  Name: "Words"
Track 4:
  channel: 1
  Name: "Melody"
  Program: Acordion
  Volume: 127  (0=mute,127=full)
  Balance: 64 (0=left, 64=mid, 127=right)
  chorus: 14 (0-127)
  reverb: 42 (0-127)
  notes: g#5 - e6 (9)
Track 5:
  channel: 3
  Name: "whistling"
  Program: Whistle
  Volume: 70  (0=mute,127=full)
```

```
Balance: 97 (0=left, 64=mid, 127=right)
  chorus: 14 (0-127)
  reverb: 42 (0-127)
  notes: g#6 - b7 (16)
Track 6:
  channel: 4
  Name: "horse running"
  Program: WoodBlok
  Volume: 90  (0=mute,127=full)
  Balance: 93 (0=left, 64=mid, 127=right)
  chorus: 14 (0-127)
  reverb: 42 (0-127)
  notes: g#4 - c#5 (6)
Track 7:
  channel: 5
  Name: "choir"
  Program: AahChoir
  Volume: 80  (0=mute,127=full)
  Balance: 30 (0=left, 64=mid, 127=right)
  chorus: 14 (0-127)
  reverb: 42 (0-127)
  notes: g#5 - e6 (9)
Track 8:
  channel: 6
  Name: "solo guitar"
  Program: Distortd
  Volume: 90  (0=mute,127=full)
  Balance: 38 (0=left, 64=mid, 127=right)
  chorus: 56 (0-127)
  reverb: 113 (0-127)
  notes: b4 - b5 (13)
Track 9:
  channel: 10
  Name: "Drums"
  Program: GM Drums
  Volume: 111  (0=mute,127=full)
  chorus: 0 (0-127)
  notes: c3 - a4 (22)
Track 10:
  channel: 12
  Name: "Bass"
  Program: FngrBass
  Volume: 95  (0=mute,127=full)
  Balance: 64 (0=left, 64=mid, 127=right)
  chorus: 0 (0-127)
  reverb: 0 (0-127)
  notes: d#2 - g#3 (18)
Track 11:
  channel: 13
  Name: "Western guitar"
  Program: SteelGtr
  Volume: 77  (0=mute,127=full)
  Balance: 85 (0=left, 64=mid, 127=right)
  chorus: 56 (0-127)
  reverb: 56 (0-127)
  notes: g#3 - c#6 (30)
Track 12:
  channel: 14
  Name: "Western Guitar 2"
  Program: NylonGtr
  Volume: 86  (0=mute,127=full)
  chorus: 42 (0-127)
  reverb: 42 (0-127)
  notes: d#4 - g#5 (18)
Track 13:
  channel: 15
```

```
Name: "Mute guitar"
  Program: MuteGtr
  Volume: 95  (0=mute,127=full)
  chorus: 42 (0-127)
  reverb: 42 (0-127)
  notes: g#3 - c#5 (18)
```

3.18 Karaoke to MIDI with lyrics



[in menu Convert]

A karaoke MIDI file contains song text in different kind of formats.
This utility converts the song into MIDI format 0 and writes the lyrics in a format that is used by different keyboard models or software players.

Choose one of following formats, try some of them if you don' t know what your keyboard supports:

- Cakewalk (4.0 or higher, META lyric commands)
- Cakewalk (3.x or higher, META marker commands)
- Tune 1000 (standard MIDI lyrics)
- Chords above lyrics (two lines for display using fixed fonts like Courier)
- Chords between lyrics (f**or all devices that can not display chords above lyrics**)
- Farfisa F5 (sysex commands)
- Roland (META lyrics)
- Yamaha XF (SL02)
- Roland sysex video lyrics
- M-Live Wordbox sysex
- Casio TV

break lines at max. characters:

Enter a maximum line length (default 40 characters) to limit the line length for certain karaoke display. Enter 0 if no limit is set (no breaking). Line breaks are inserted when song text lines are longer than this limit, in most cases the line break is done at a space, in rare cases it is done within word (e.g. when no space near end of limit).

options for chords between lyrics conversion:

move chords in front of words
moves the chord names that are inside a lyric word to beginning of the word that the word can be better read.

remove chord identifiers (like [...])
removes text chord symbols [Cm] {Cm} (Cm) %Cm "Cm"

enclose chords by <...>
adds < and > around the chord names that the chords can be easier distinguished from lyrics e.g. <Cm>

add spaces before and after chord names
adds a space before and after the chord name that the chord names do not collide with other near lyrics e.g. dis Cm play instead of disCmplay

Try different formats for use with your keyboard and choose the best one, start with option Tune 1000 that works with many keyboard displays. Some keyboard displays don't accept longer lines, specify a maximum line length that GNMIDI automatically breaks the lines into smaller ones.

.kar format
.kar files are MIDI file format 1 files that contain META text commands to store the text in a special formatting.
This operation can convert .kar files into other lyric formats.
Use Convert MIDI with lyrics to Karaoke MIDI to convert a MIDI file to a MIDI file with .kar formatted lyrics.

Hint: After converting to chords between lyrics format the chords can not be edited using chords editor since this function generates text only. Use the edit words operation to modify the chord names text.

3.19 Convert MIDI with lyrics to Karaoke MIDI

[in menu Convert]

This operation converts a MIDI file that contains song text into a **.kar** compatible MIDI file. Save the files with file extension .kar to play them with other karaoke players that support .kar.

The conversion creates a format 1 MIDI file that contains .kar formatted META text commands (line breaks, paragraphs, karaoke info).

3.20 Trim mute song ending

[in menu Modify]

This operation deletes a pause (that is longer than a second) at end of MIDI song (only pause that is behind end of last playing note).

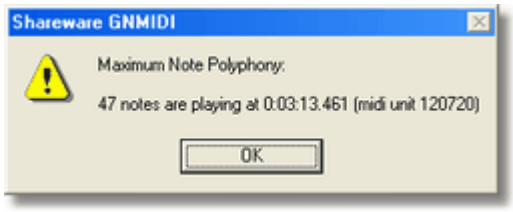
Hint: If a MIDI song contains notes that are not stopped correctly (hanging notes) or if a long note at

end of song becomes silent before it is really stopped then this operation is not available, because there is not really a pause at end of the song (there are still notes not turned off). Use cut operation to get copy a part of the MIDI song into a new MIDI file.

Hint: The silence at end of the song must be longer than a second that this operation deletes the pause.

Hint: This operation is available as batch operation. Using GNMIDI Light license batch operations are not available.

3.21 Calculate maximum note polyphony



[in menu Analyse]

This operation calculates the maximum number of notes playing at same time (considering influence of used piano pedals).

It displays a message box with following info:

- number of notes playing at same time
- time position where this number of notes are playing (hour:minute:second:millisecond)
- MIDI position where this number of notes are playing (in MIDI units)

Notes that are not stopped correctly can increase the calculated maximum note polyphony. If the value is high and the position is near end of song then you should check if the song contains hanging notes.

Hint: open logfile to find the list of notes playing at the position with maximum polyphony (at end of file).
Hint: pedal may hold notes that already received note off till hold off is received. With sounds like piano this cannot be noticed (piano automatically gets silent after some time).
Use sound like organ that play till turned off correctly.

3.22 Reset GM, GM2, GS, XG, INIT

[in menu Player/Reset MIDI device]

These commands play

- a common used standard sysex command (default)

or

- a user defined MIDI file (one for each button).

It is useful to reset the sound card or force it into a certain mode before playing MIDI files, otherwise certain MIDI devices can react unexpectedly by playing wrong sounds (e.g. drums play piano sound). The standard sysex commands will work only if your sound device supports GM, GM2, GS or XG compatible modes.
Read about supported MIDI commands in MIDI implementation chart of your keyboard manual.

INIT
by default INIT button sends MIDI commands that reset common used controllers, pitch bend, sound programs.

You can redefine the buttons for playing your own initialization MIDI files by modifying GNMIDI.INI file:

```
[Settings]
GM=drive:\path\filename.mid
GM2=drive:\path\filename.mid
GS=drive:\path\filename.mid
XG=drive:\path\filename.mid
Init=drive:\path\filename.mid
```

Replace the right sides of = by path of an existing MIDI file, e.g.

```
GS=c:\gnmidi\gsreset.mid
```

The content of your user defined initialization MIDI files is not limited by any rules, you can even play notes in such a file.

3.23 Reset device (SOS)

[in menu Settings]

SOS command sends few MIDI commands to current output device in order to stop playing notes, reset controllers and settings that future songs will start playing at a defined state.

This command is usually used, if a currently stopped song did not stop all notes.

Hint: This command can be set as player option, so that it SOS will be automatically done after each stop of player.

3.24 Prepare MIDI song before playing

[in menu Settings]

These options prepare a MIDI file before it is played with internal or standard MIDI player, selecting one or more options will play a temporary MIDI file instead of the original MIDI file.

No mode changes
plays the original MIDI file without any initialization or program modifications, except common volume option that will be considered.

GM only
removes all non-GM MIDI commands (bank controls, sysex) and add GM Reset sysex. This option is useful for GM compatible MIDI devices that would play GM2, GS or XG MIDI files wrong.

GM reset
adds a GM sysex.

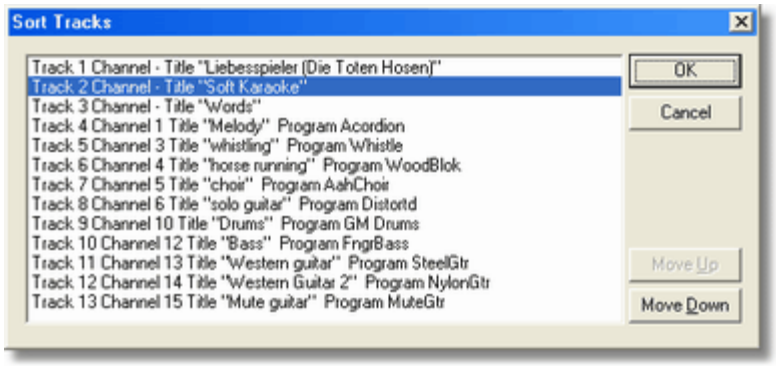
GM2 reset
adds a GM2 sysex.

GS reset
adds a GS sysex.

XG reset
adds a XG sysex.

Common volume
adjust volume level to a common volume level by modifying some controller values.

3.25 Sort tracks in a format 1 MIDI file



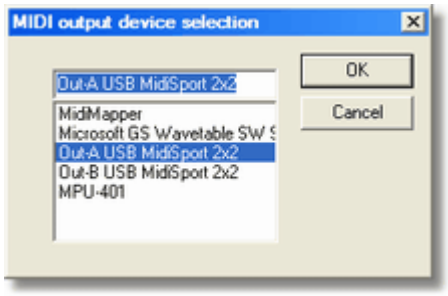
[in menu Modify]

This operation is used to change the current order of tracks in a MIDI format 1 file. This is useful when you need a track at certain position or when your MIDI device can only play few tracks.

The dialog shows current tracks of a MIDI song. Each line contains:
- Original track number: Track 1 etc.
- Track channel number: 1-16 or - (if no channel MIDI events) or multi (if more than one channel used in this track)
- Track title: Title "..." or Tempo track or none
- Track initial program: Program ... or none

Move Up
Move Down
select a line and move it up or down. It is not allowed to move track 1 (Tempo track). It is not allowed to move a track in front of Tempo track. It is not possible to move last track down. The buttons are disabled in these cases.

3.26 Select MIDI output device for internal MIDI player

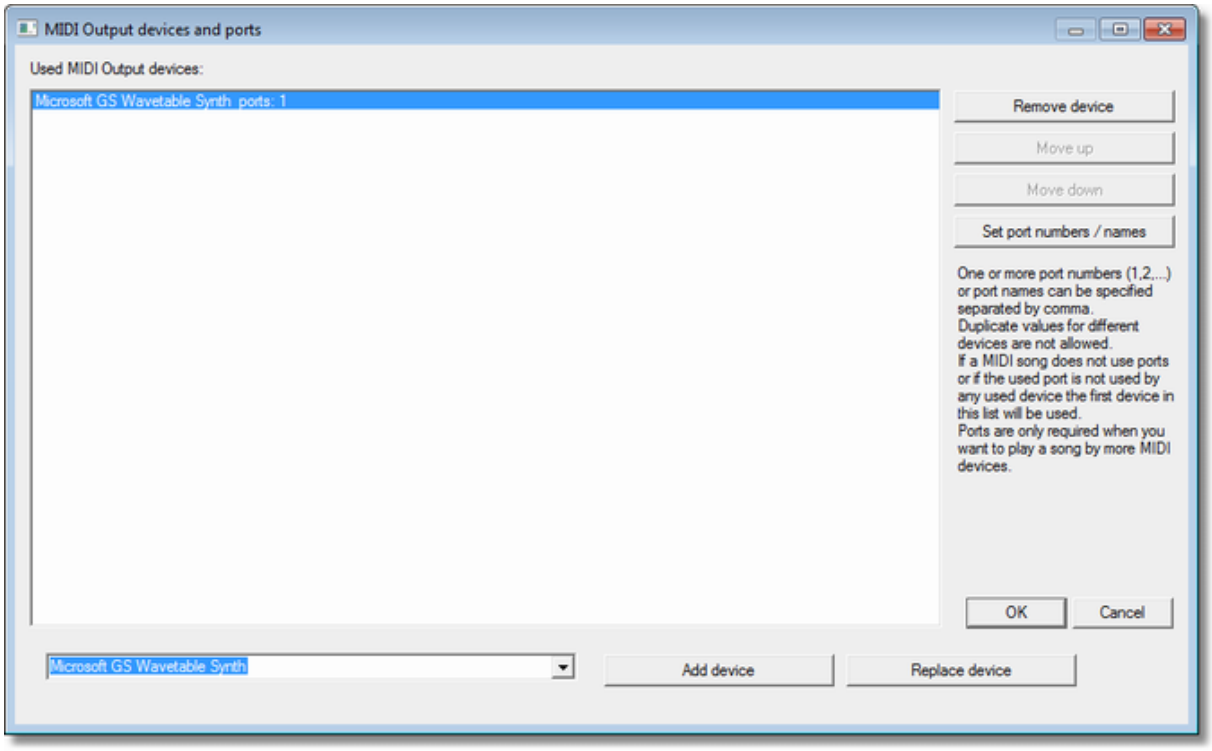


[in menu Settings]

Windows system allows to install more than one MCI compatible MIDI devices (sound card, software synthesizer, external MIDI device connection, ...).
GNMIDI internal player plays MIDI files through the device selected from this list of output devices.
This setting does not affect external MIDI players, they should have their own settings.

MidiMapper
Choose MidiMapper if you want to use the device that is used by Mediaplayer as standard output device.

3.26.1 MIDI output devices and ports



Play a MIDI song using more MIDI devices parallel

It is possible to play a song through MIDI cables by more than one MIDI device.
Each device could use up to 16 MIDI channels.
The MIDI song needs to contain META port number or META devicename / portname commands (usually at beginning of each track).

Select more MIDI devices for MIDI output
Choose some MIDI devices and assign optional a list of port numbers and port names to each device.

Move up, Move down
This changes the order of the devices in the list. The first device is used as default device if a song does not use ports or uses ports that are unknown.

Port numbers (1-256)
are in META port command counted 0-255. The MIDI standard has declared using port numbers out of date. Many applications still use numbers.

Port names
MIDI standard calls them device names. In GNMIDI more port names can be used for one device in case that different songs use other port names and mean the same device.
Matching port names have in GNMIDI and GNMixer more priority than matching port numbers.
GNMIDI ignores the case of the port names

Enter a list of numbers or names
The list of names will be separated by commas e.g. 3, Yamaha, Yamaha CVP-305

Default handling
If a MIDI song uses a port number or port name that is not found in the assigned ports of chosen output MIDI devices then the first MIDI device in the chosen output devices is automatically used as default.
If only one device is chosen then surely only this device will be used independent of the MIDI ports in the song.

MIDI modes and reset settings
The MIDI song must contain all settings necessary for the different devices. If not all devices use same MIDI mode (e.g. GM compatible) then the player setting should use no mode and the MIDi song needs to do the initialisation properly.

Latency problem
Latency is the delay time between sending a MIDI command to a device till hearing the sound.
Good MIDI devices react so quick (no or minimal delay between sending a command till hearing the sound) that there is practically no latency.
If MIDI devices react slower than other MIDI device then playing them parallel a song then the delay can be heard awefully.
For playing songs parallel to more devices use MIDI devices with no latency problems.

3.27 Compress MIDI

[in menu Convert]

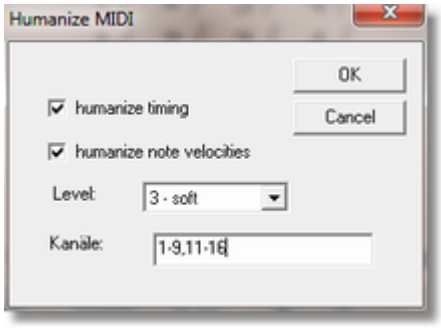
This operation tries to compress your MIDI file. The compression does not change your MIDI song content, it only stores the data more compact if possible. After compression some statistics on compression are displayed. The compression ratio depends on input, the results can be 10%-15% smaller than original file size.

A compressed file can't be compressed again, in this case the operation will tell
```
Compression not necessary
```

Invalid files can't be compressed:

```
Operation failed
```

3.28 Humanize MIDI



[in menu Modify]

Use this function to make computer sequenced or generated MIDI songs with exact timing more like played by human, with little inexactness in timing and key pressure. Try different levels to find a middle between too exact and too inexact.

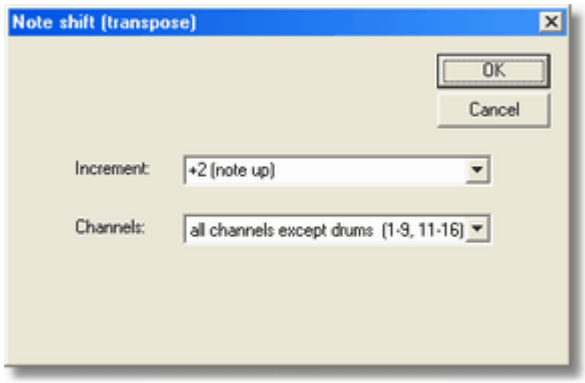
humanize timing
produces little random timing errors within the selected level. Timing is limited to remain within 1/16 note of original timing. This affects the position of all commands and notes and duration of notes.

humanize note velocities
varies all note velocities by random within selected level. This affects the key pressure of note on/off commands, usually this influences notes volume and sound effects.

Level of humanizing
choose a level between 1 and 10, where 3 is soft, 5 is medium, 10 is heavy.

Channels
a range list of channel numbers 1-16

3.29 Transpose MIDI



[in menu Modify/Note operations]

Use this operation e.g. if you need to transpose song notes to the song key of the lead singer. Transpose operation (also called shift note) transposes note values some half tones up or down.

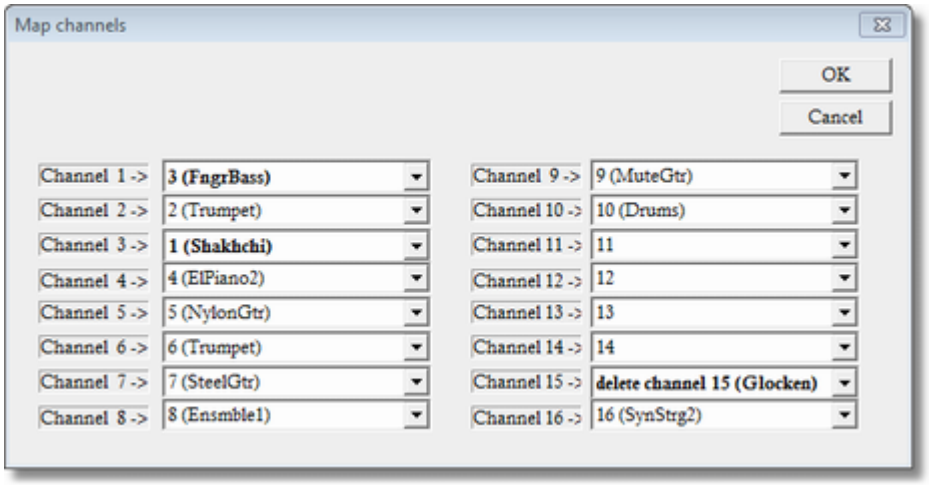
Increment
Enter a increment number of half tones between -12..+12. Negative values shift notes downward, positive values shift notes upward. 12 half tones are an octave.

Channels
Transpose can be applied to
- all channels (1-16)
- all channels except drums (1-9,11-16 assuming GM drum channel 10)
- single channel (1-16)

Usually the operation is applied to all channels except drum channel.

Hint: This operation is also available as batch conversion. Using GNMIDI Light license batch operations are not available.

3.30 Map Channel Numbers



[in menu Modify]

This utility renumbers the channel numbers of all MIDI commands. Fill the mapping table for each channel that you want to renumber.
E.g. map channel 1 to 3 and channel 3 to 1 will exchange current channels 1 and 3.

Hint:
You can map one or more channel numbers at once (exchange their numbers). If more track use same channel number then changes are done in all these tracks.

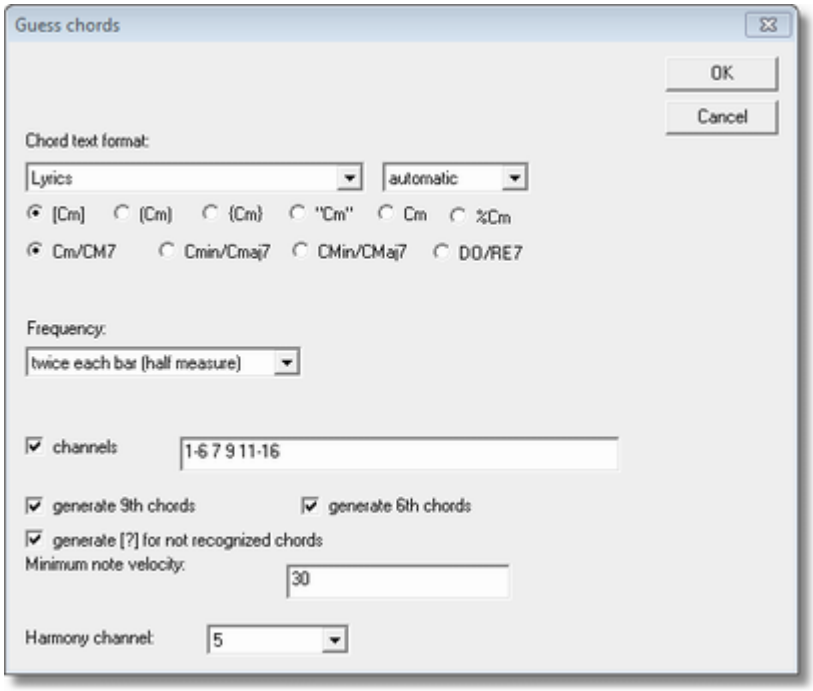
Hint: Inside parentheses () the first **sound name** from the original channel will be displayed if the channel is used.

Hint: select **Delete channel** to remove all MIDI commands on this channel

This operation is also available for batch application. Using GNMIDI Light license batch operations are

not available.

3.31 Guess Chords



[in menu Analyse]

This operation analyzes chords for a MIDI song. It is called "guess" chords because in many situations there are more than one possible chords and an automatic analysation must decide for probably best chord.

The chord text format defines how the resulting chord info should be stored inside the resulting MIDI file. Choose the MIDI commands that are to use and how the chord names should be formatted as text.

Chord output

- Markers
- Lyrics
- Text
- Harmony notes (produces chord notes instead of named chords)
- Yamaha PSR meta (XF)
- M-Live Wordbox sysex
- Pgmusic Inc. Band-in-a-Box
- GEM META (Generalmusic, e.g. GEM WK4, GEM GENESYS)
- Ketron META (e.g. Ketron SD8, Ketron Audia)

Sharps or Flat keys

- use sharp keys (#) e.g. F# is preferred instead of Gb
- use flat keys (b) e.g. Bb is preferred instead of A#

Chord text formatting

- [Cm]
- (Cm)
- {Cm}
- "Cm"
- Cm
- %Cm (used by Technics KN keyboards)

Chord name formatting

- Cm/CM7 short chord names
- Cmin/Cmaj7 long chord names beginning with lower case letter
- CMin/CMaj7 long chord names beginning with upper case letter
- DO/RE7 italian chord names (DO, RE, MI, FA, SO, LA, SI)

Frequency

- Each bar (measure)
- twice each bar (half measure)
- each beat (quarter note)

Hint: It is important that the bar information in the MIDI file is correct else notes start at random positions within a bar and that causes inexact results!
Hint: analyzing chords for each bar usually gives better results than doing it for each beat.

Harmony channel

is only used for chord output **Harmony notes**, the generated chord notes will be added to this channel. Harmony notes are used for controlling a vocal harmonizer.

Channels

option checked: a list of channels can be specified e.g. 1 2 3 5-9 11-16
option unchecked: default channels 1-9 11-16 are used

By default all channels are considered for analyzing (melody, bass, accompaniment, ...) together. Only the notes matching the given channel list will be used for analyzing chords. This might give better results if the MIDI song already contains a chord notes track (specify the channel of these chord notes).

Hint: channel numbers are 1-16
Hint: only specify non-drum channels
Hint: channel 10 is in most cases a drum channel

Recognized chord types

- Major chords
- minor chords
- sus4
- 9th
- dim
- aug
- 6th
- 7th
- 7Maj

Hint: **Missing notes** can cause that a chord is not found, e.g. C-G-A won't be recognized as C6 chord since note E is missing.
Hint: If chords are not recognized (or wrong) because of missing notes then you could add the complete chord notes in a new track (silent) and repeat chord guessing.
Hint: **pitchbending** is currently not considered. If a note is changed by pitchbend near to an other note

still the specified note is used for recognition.
Hint: since the chords are generated for a bar, half bar or beat the **bar information is essential** for this operation. If bar information is incorrect then notes from previous bar section and next part section would be used and the mixture of notes cause other and probably wrong recognized chords.
The text events are synchronized to the bar positions. Some MIDI players can show them while playing (e.g. midimach, GNMIDI, GNMixer).

Use 9th chords
if checked chords like Cm9 might be generated

Use 6th chords
if checked chords like Cm6 might be generated

generate [?] for not recognized chords
If too few notes are available within a bar then the analysation won't guess the chord and the missing chord will be marked with ? (e.g. [?]).

Minimum note velocity:
ignore all notes that have a note on velocity lower than given minimum value. Using 0 or empty input will consider all notes matching the channels list.

Wordbox
Wordbox is a MIDI text player from Italian producer M-Live, which transfers song text and chord names through MIDI cable and mainly supports musicians on stage during singing.
Instead of exporting guessed chords directly into a Wordbox compatible format, you can first merge the chord text enclosed in [...] with song text (choose lyric or text depending on the current used song text format) and later convert song text including chord names into a Wordbox compatible format. This has advantage, that the analyzed chord names can be checked and modified if necessary, which can't be done after exporting them into a Wordbox sysex format using GNMIDI.

Hint: Most chord are stored as readable text, they could be modified with operations like modify text
Other formats are very special for a certain device and can not be modified with GNMIDI. Chords in format Yamaha PSR meta could be modified with GNMIDI using conversion to ASCII text and back (there the psr chords look like psrmeta chord "Am").

Band-In-The-Box™
is a famous MIDI Software by company Pgmusic Inc. which uses chords as a key feature during work with MIDI files. The generated chords usually can only be used with the software products of this company.

Following setting in gnmidi.ini can be used to force generating of empty chord lead measures:

```
[Settings]
BIABLeadMeasures=0
```

Hint:
GNMIDI shows chords of some known chord formats above the song text line and also displays chords using operations copy and print song lyrics.

3.32 Split Drums

[in menu Modify/Sound operations]

General MIDI drums are always at channel 10. Each note is an other drum instrument. This operation generates a new track for each used drum instrument and moves the notes into the corresponding track. Each generated drum track is named by the GM drum instrument.

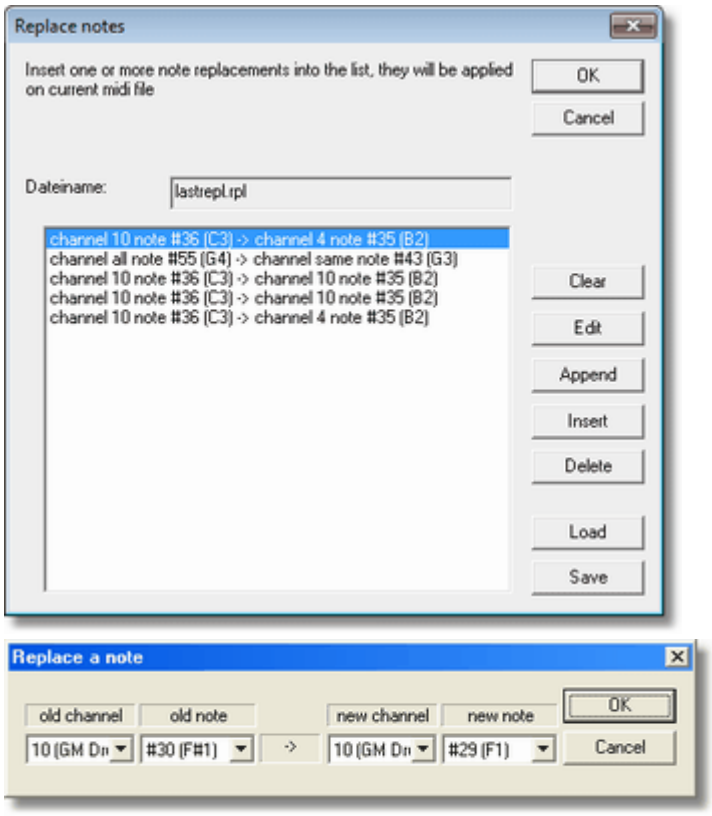
When loading such a file into a sequencer then the drum tracks have separate lines for each drum instrument, which is sometimes easier to edit or understand.

3.33 Split Programs

[in menu Modify/Sound operations]

This operation separates all channels that contain more sound program changes into own tracks (one for each sound program).
The channel number and sound does not change, only the display changes when loading the file into a sequencer.

3.34 Replace Notes



[in menu Modify/Note operations]

This operation replaces notes numbers that match the given channel number by new channel and new note number. The translations are entered in a note replacement table which can be stored and reloaded for future sessions. The table remembers its last content. Each row in the table defines a note replacement. Replacing is done in order of table entries.

Hint: drum channel contains drum notes where each note number is an other drum instrument. This operation can be used to convert drum tables between two devices (e.g. keyboard drums to GM drums)

Hint: set old channel to "any except GM drum channel" if you want to replace or delete notes on channels 1-9,11-16.

Hint: set old channel to "any" if you want to replace or delete notes on all channels.

Hint: set new channel to "same" if you don't want to change the channel number against a new channel number.

Filename
displays the last loaded file name. A * sign is appended when the data has been modified since loading. If exiting the dialog with OK button and data has been modified without saving then the data is automatically stored in lastrepl.rpl file.

Clear
start a new replacement table.

Edit
edit the current selected replacement in an own dialog. Double click with left mouse button on the line does same.

Append
add a new note replacement at end of table.

Insert
enter a new note replacement before current selected row.

Delete
deletes current selected row

Load
Previously saved replacement tables (*.rpl files) can be reloaded. You can also edit *.rpl files with notepad, they are simple text files.

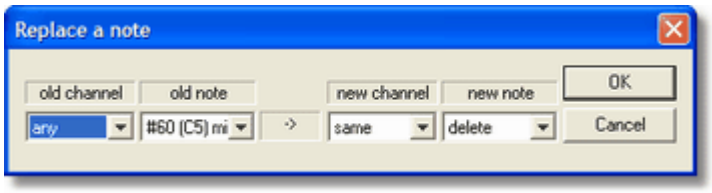
syntax:
```
oldchannel:oldnote -> newchannel:newnote

where oldchannel can be 1-16 or -1 for any channel or -2 for any channel except
drum channel 10
oldnote can be 0-127
newchannel can be 1-16 or -1 for same channel
newnote can be 0-127 or -1 for delete note
```

Hint: oldchannel -2 is available with GNMIDI v3.33

more complex mapping files can be written by programmers using scripts (e.g. perl, gnscript)

flip notes
load flipnotes.rpl that replaces note 0 against 127, 1 against 126 ... on all channels except drum channel 10
load flipnotesinsameoctave.rpl that replaces all C against B, C# against A# ..... on all channels except drum channel 10
Hint: these *.rpl files can be found in GNMIDI program folder in newer GNMIDI versions
Hint: the results could sound randomly nice but do not necessarily

Save
Save your replacement table in a .rpl file that the replacement can be used later again for other MIDI files.

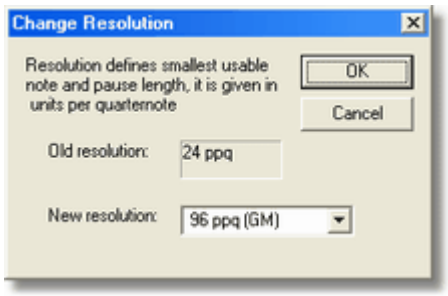


Deleting notes
This operation is also usable for deleting certain note numbers from given or any channel by setting new note to value "delete".

Moving notes
This operation is also usable for moving certain note numbers from one channel to a new channel. To renumber a whole channel you should use operation Map channel numbers.

This operation is also available for batch application. Using GNMIDI Light license batch operations are not available.

3.35 Change Resolution



[in menu Modify]

MIDI resolution can be changed without changing tempo, the pauses are quantised to the new resolution steps.

Old resolution
current MIDI song resolution

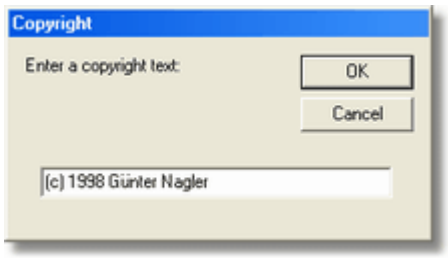
New resolution
choose new resolution value from a list of common used resolution values.

MIDI Resolution (ppq)
MIDI resolution defines number of units per quarter note for whole song (resolution). A MIDI unit is the smallest usable pause or note duration in the song. The smallest usable pause or note is 1/(4*resolution) units (e.g. for resolution 96 the smallest note is 1/384 = 1/(3*128), that means that it can be used precisely for trioles of 1/128 notes).

Hint: 96 ppq is recommended for General MIDI compatible songs.

Hint: Some players might not be able to play songs with too high resolution, reducing resolution with this operation should help.

3.36 Set copyright information



[in menu Modify]

If the current MIDI document **contains a copyright notice** then the copyright is displayed (the info is also displayed in the document window).

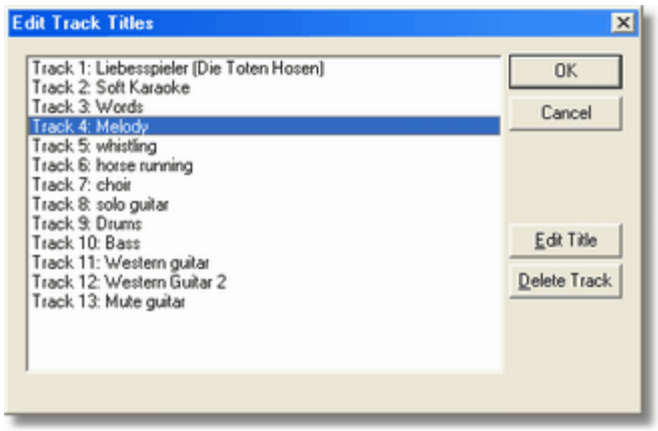
```
Copyright:
(c) 1990 Die Toten Hosen
```

If **no copyright** command is in the file then the operation allows to enter a new copyright text. The new copyright notice is added to first track.

It is not possible to modify an existing copyright with this operation!

Legal notice:
Add copyright only if you are owner of copyright, i.e. you are composer of the song or you have licensed the distribution rights for this song from the copyright owner.

3.37 Edit track titles



[in menu Modify]

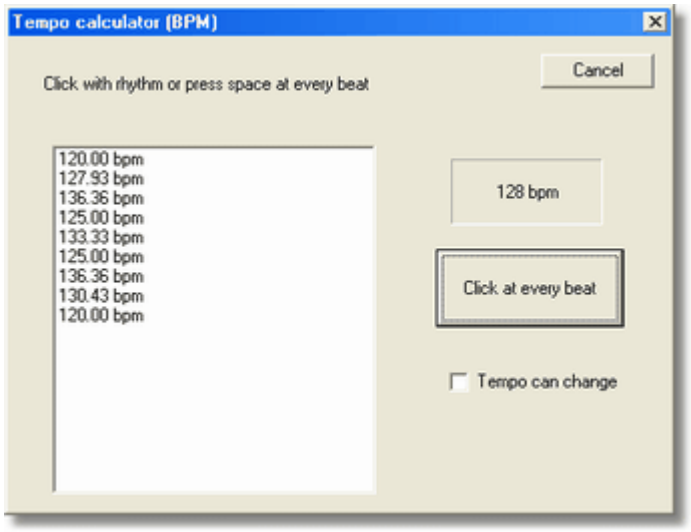
This utility is used to rename track titles (also called track names) or optionally delete a track. Select a track before editing or deleting a track.

Edit Title
Current selected track title will be displayed and can be replaced by a new title.

Delete Track
Marks track title of selected track with `!delete!` `With OK button all marked tracks that way will be deleted.`

Hint:
if a track contains more than one track titles (should not occur in a standard MIDI file) then all old track titles are removed and the new track title will be inserted at beginning of the track.

3.38 Calculate tempo (BPM)



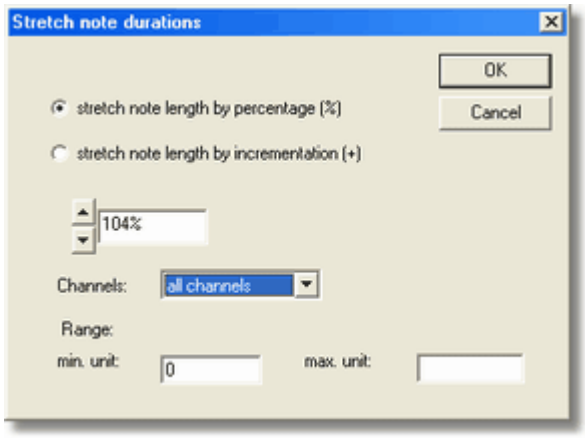
[in menu Analyse]

This tempo calculator is used to find out tempo of an existing song on CD or tape. The dialog shows the calculated tempo in **bpm** (beats per minute)

Click at every beat
Click left mouse button or press Space key at every beat in the song. Try to hold the speed of the rhythm. The average delay between clicks will be calculated into bpm tempo values. For constant tempo the calculated value gets more and more precise when you click longer in same tempo.

Tempo can change
By default the calculator assumes that the rhythm plays at constant tempo. Check the option if the tempo can change during the song.

3.39 Stretch notes duration



[in menu Modify/Note operations]

This operation increases or decreases note durations by specified percentage or value.

stretch note length by percentage (%)
enter a percentage value, all note durations in given range and channels will be stretched by this percentage.

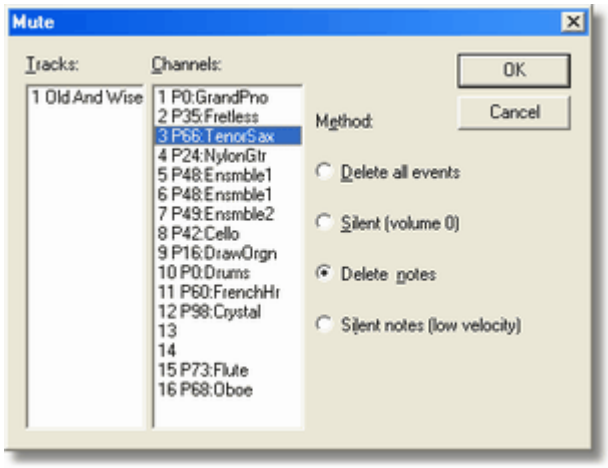
stretch note length by incrementation (+)
enter a value, this value will be added to all note length (MIDI units), negative values will decrease note length.

value
enter a value into the edit field or use the arrows to increase or decrease the value

Channels
choose all channels (1-16, default) or select a single channel (1-16)

Range (min. unit - max. unit)
by default the operation will be applied to notes of any length, in the two range value fields you can define that the note length must be in given range (MIDI units) that the operation should applied (e.g. min=0 max=5 would modify only very small notes). If the max value is missing then no max value is defined (min .. any higher value)

3.40 Mute voices



[in menu Modify/Volume operations]

This operation mutes a MIDI voice by one of four available methods. This is useful to remove parts that real musicians play live in a band or for muting voices that are replaced by karaoke singers.

Tracks:
Choose track if you want to mute all channels in a certain track or if you want to mute a certain channel only in one track.

Channels:
Select one or more channels that should be muted.

Hint: If you select only tracks then any channels on this track are used for mute operation.
Hint: If you select only channels then any tracks that contain these channels are used for mute

operation.
Hint: If you select tracks and channels then only those tracks are used that contain one of the given channels.

Method:

- Delete all events — deletes all commands on the matching channels and tracks
- Silent (volume 0) — sets volume of matching channels to 0
- Delete notes — deletes only the notes on the matching channels and tracks
- Silent notes (low velocity) to a low value (1) — sets velocity values of the notes on the matching channels and tracks

Hint: Silent (volume 0) method is useful if you want to sing a muted melody (karaoke) and still need the notes displayed on screen with a sequencer or a special MIDI player.

Hint: Delete notes method is useful if you want to play these deleted notes self with original settings.

Hint: The operation can't be undone so you need to keep your original MIDI files if you want the muted tracks back.

Hint: With the option Remove all commands you can delete a channel completely.

3.41 Generate setup measure

[in menu Modify]

This operation inserts a new tact bar (measure) with optional initialization commands.

- GM Reset — initialize General MIDI compatible devices
- XG Reset — initialize Yamaha XG compatible devices
- GS Reset — initialize Roland GS compatible devices

Following MIDI parameters are initialized to GM default values if the parameters are not initialized before playing first note in track:

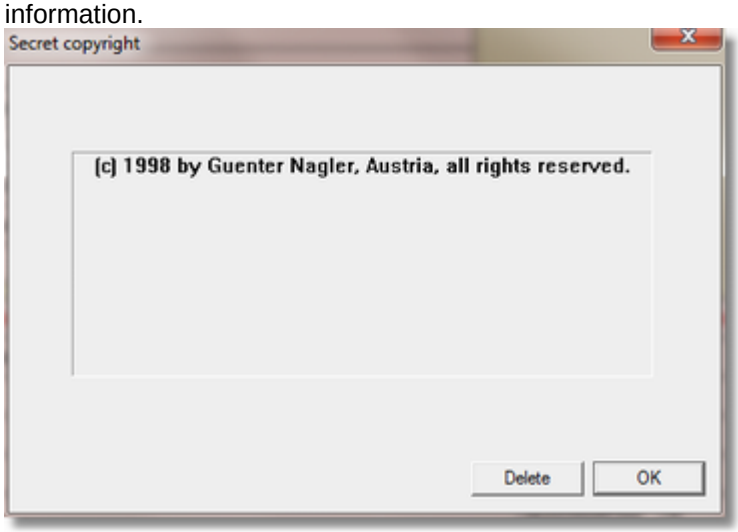
- volume controller: 100
- balance controller: center
- program: Piano (except drum channel 10)
- pitch bend: center (no pitch bending)
- pitch bend range: +/-2 halftones (with RPN commands, not supported on all devices)

3.42 Show or add secret copyright

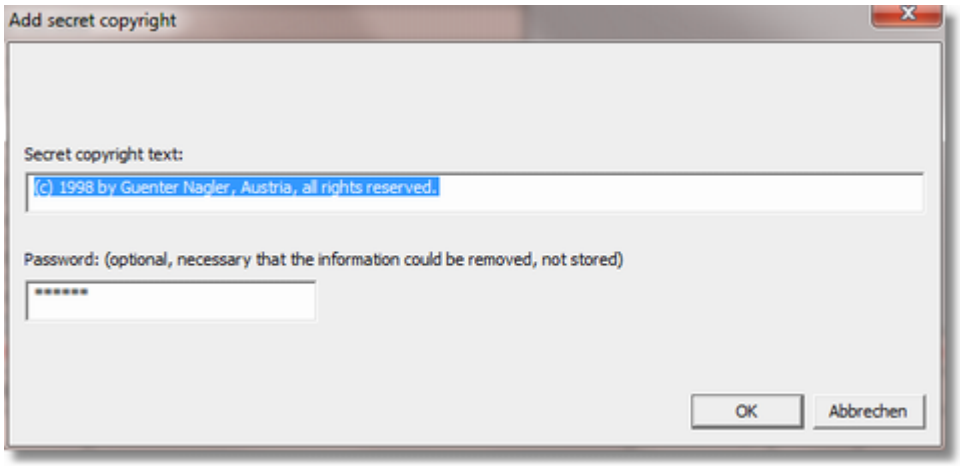
[in menu Modify]

This function adds encoded copyright information to the MIDI file. In cases that somebody illegally removes or changes the standard visible copyright field, the secret copyright information remains in the file hidden. The secret copyright can prove your copyright when somebody else tries to steal it.

If a secret copyright (done with GNMIDI) already exists then it **displays the secret copyright** information.

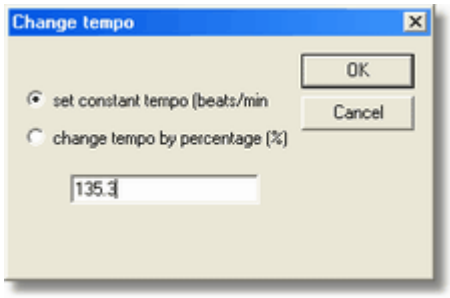


If no standard and no secret copyright found then program asks for a copyright line. Use only ASCII characters (characters blank - ~), other characters (like international characters e.g. ü) are not stored. An optional password could be entered which can be used to delete the secret copyright text later. You must remember the password. The entered password is not stored in the MIDI file. You must remember the password.



Delete secret copyright
The show secret copyright dialog contains a delete button. You must enter the password that was used during entering adding the secret text.
If no password was used or password forgotten then you need to use your midi file backup.

3.43 Set tempo (bpm and percentually)



[in menu Modify/Tempo operations]

This operation changes tempo of a MIDI song by two different ways.

- **set constant tempo (beats/min)**
  enter a number of beats per minutes (bpm) between 40 and 240 in the edit field below. All existing tempo changes in the MIDI file are removed and the new tempo change will be inserted at beginning of the song.

- **change tempo by percentage (%)**
  enter a percentage value between 1 and 200 in the edit field below. All existing tempo changes in the MIDI song are changed by the given percentage. A new tempo change might be inserted at beginning of the song if the song used default tempo 120 bpm at beginning of song (no tempo change at start position)

edit box
Enter a number in this edit box. Floating point values are allowed (e.g. 100.3). The radio options above determine the meaning of the value (bpm or %)

3.44 Check all MIDI files

[in menu Analyse]

This batch operation checks all MIDI files for being valid in specified MIDI folder. At completion a list of errors is shown and you are asked if you want that GNMIDI tries to repair the files. Repaired files will be opened in a document window. You need to check if the repairing is acceptable and then save the MIDI file.
Using GNMIDI Light license batch operations are not available.

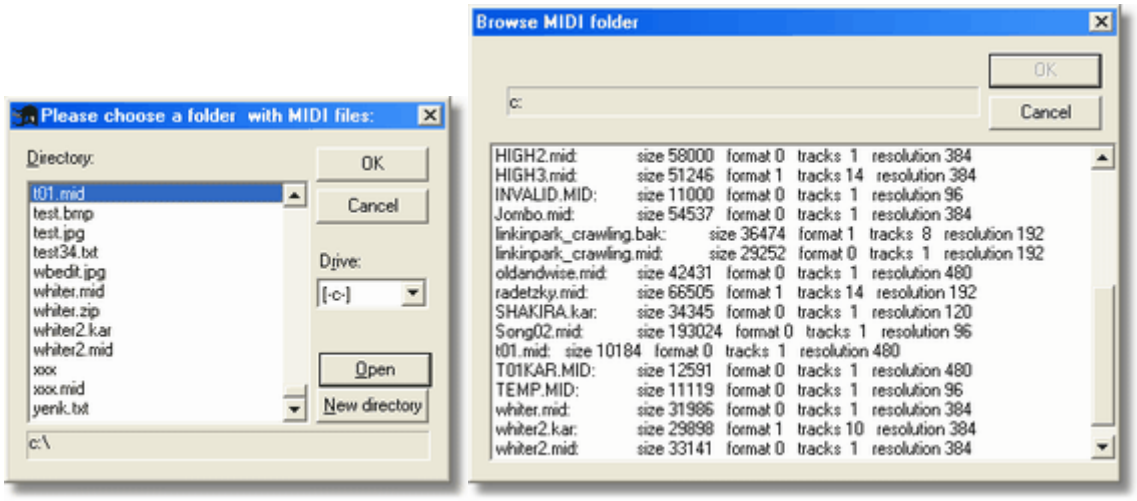
Checking same folder again will work quicklier, since it won't check those files again which didn't change. If you check all files in this folder again then you can delete the file with extension .chk

3.45 Create new MIDI file

[in menu File]

This operation creates an empty format 1 MIDI file with some initializations and one measure without notes. You can use it as start for sequencing a new song.

3.46 Browse MIDI folder



[in menu File]

The browse operations displays folder content. It displays basic MIDI file information and can be used to select one or more MIDI files at once for opening. The operation uses information that is generated by check all MIDI files operation (*.chk) to load information quicklier if the check operation was used previously. Selecting more files at once can be done with combinations of Ctrl, Shift and arrow keys.

This dialog is also used by most batch operations.

3.47 Reverse MIDI song

[in menu Convert]

This operation mirrors position of notes within current MIDI song. This gives possibility to play a song reverse. The mirrored result can be reversed again but it might be slightly different to the original MIDI song.

A **Mp3 file** can also be reversed if FFmpeg package is installed.

3.48 Check midi natural instrument note ranges

[in menu Analyse]

The operation checks if notes played with natural sounds (e.g. trumpet, violin) are within a **realistic note range**. Too high or too low notes might cause to play this sound in an unrealistic way.
The note range definition file for a certain device (usually the text file gm.rng which contains note ranges of some General MIDI standard instruments) will be loaded. All notes in current MIDI file are checked against the note range definitions and a warning is reported if a note is outside the range specified in the range definitions for the instrument which plays the note.

The note range definition file can be modified or a new definition file defined and activated by changing gnmidi.ini file :

```
[Settings]
   RangeDefinitions=myranges.rng
```

The range definition file should be in GNMIDI installation folder, else a full path must be specified.

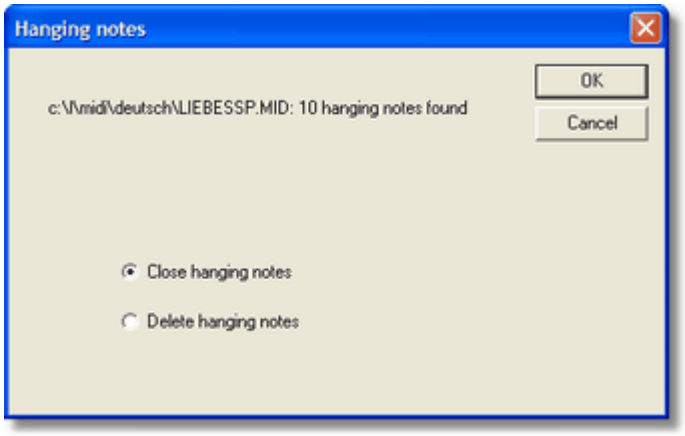
When all notes are in a valid range then following message is displayed:

```
  All notes are in range (according to note range definitions)
```

When some notes are in an invalid range then the information is displayed in a text file with notepad editor:

```
  checking note ranges according to  gm.rng:
  track 3 channel 2 measure 3 Violin (40) note f3 range g3-c7
  track 3 channel 2 measure 3 Violin (40) note e3 range g3-c7
  track 3 channel 2 measure 3 Violin (40) note f3 range g3-c7
  track 3 channel 2 measure 3 Violin (40) note e3 range g3-c7
  track 3 channel 2 measure 4 Violin (40) note c3 range g3-c7
  track 3 channel 2 measure 7 Violin (40) note f3 range g3-c7
  track 3 channel 2 measure 8 Violin (40) note e3 range g3-c7
  track 3 channel 2 measure 8 Violin (40) note c3 range g3-c7
  track 5 channel 4 measure 98 Ovrdrive (29) note d2 range e2-d6
  track 5 channel 4 measure 101 Ovrdrive (29) note g#1 range e2-d6
  track 5 channel 4 measure 101 Ovrdrive (29) note b1 range e2-d6
  track 5 channel 4 measure 101 Ovrdrive (29) note c#2 range e2-d6
  track 5 channel 4 measure 101 Ovrdrive (29) note d#2 range e2-d6
  track 5 channel 4 measure 101 Ovrdrive (29) note c#2 range e2-d6
  track 5 channel 4 measure 101 Ovrdrive (29) note b1 range e2-d6
```

3.49 Remove hanging notes



[in menu Analyse]

This operation checks if current MIDI file contains notes that are not stopped (a note on command is missing note off command). It considers pedal commands hold and sostenuto and all-notes-off. When such notes are found then the operation offers to stop those notes or remove those notes.

Hanging notes cause that polyphony increases unnecessarily and depending on the used sound it can be heard. The notes are on till end of song, some sounds are fading out earlier so that the problem might not be noticed. Such notes are usually unwanted bugs in MIDI files.

3.50 Delete midi tracks

[in menu Modify]

This operation starts same dialog as edit track titles operation.

3.51 MIDI to ASCII Text

[in menu Convert]

This operation converts a MIDI file into a readable ASCII text that represents the MIDI content. You can edit this text file with a text editor (Notepad editor). The (modified) text can be converted back to a MIDI file.

Here you find some information about the used grammar.

Here is a part of the generated text:

```
mthd
  version 1 // several tracks with separated channels to play all at once
  // 8 tracks
  unit 96 // is 1/4
end mthd

mtrk  // track 1
 /* U0 */  /* 0ms */   beats 142.38051 /* 421406 microsec/beat */
 /* U0 */  /* 0ms */   trackname "Gasp For Breath (G\xfcnter Nagler, 23.2.1999)"
 /* U0 */  /* 0ms */   text "This song is composed and sequenced by Günter Nagler."
 /* U0 */  /* 0ms */   text "Freely distributed for personal, non-commercial use
only."
 /* U0 */  /* 0ms */   text "Contact: info@gnmidi.com"
 /* U0 */  /* 0ms */   tact 4 / 4 24 8
  536/4; /* U51456 */  /* 225873ms */
end mtrk

mtrk(1)  // track 2
 /* U0 */  /* 0ms */   trackname "More strings at Refrain"
 /* U0 */  /* 0ms */   program Ensmble1
 /* U0 */  /* 0ms */   volume 127
 /* U0 */  /* 0ms */   balance  46
 /* U0 */  /* 0ms */   reverb 52
 /* U0 */  /* 0ms */   chorus 65
  7765; /* U7765 */  /* 34085ms */ +a4 $58;
  42; /* U7807 */  /* 34269ms */ -a4 $40;
  11; /* U7818 */  /* 34318ms */ +a#4 $52;
  44; /* U7862 */  /* 34511ms */ -a#4 $40;
  3; /* U7865 */  /* 34524ms */ +a4 $52;
  47; /* U7912 */  /* 34730ms */ +g4 $4E;
  5; /* U7917 */  /* 34752ms */ -a4 $40;
  25; /* U7942 */  /* 34862ms */ -g4 $40;
  19; /* U7961 */  /* 34945ms */ +f4 $4E;
  92; /* U8053 */  /* 35349ms */ -f4 $40;
  3; /* U8056 */  /* 35362ms */ +e4 $58;
  94; /* U8150 */  /* 35775ms */ +c5 $52;
  12; /* U8162 */  /* 35828ms */ -e4 $40;
  11; /* U8173 */  /* 35876ms */ -c5 $40;
  27; /* U8200 */  /* 35995ms */ +c5 $4A;
  26; /* U8226 */  /* 36109ms */ -c5 $40;
  26; /* U8252 */  /* 36223ms */ +a#4 $46;
  35; /* U8287 */  /* 36376ms */ -a#4 $40;
...
end mtrk
```

3.52 ASCII Text to MIDI

[in menu Convert]

This operation converts a text generated by operation MIDI to ASCII text back to a MIDI file, if the text has valid syntax. The text can be modified by the user, but must not contain syntax errors that it can be converted to MIDI.

3.53 ASCII Text syntax

MIDI unit position and bar position (measure.beat.tick) and time position (millisecond) are shown in comments before each command:

```
/* U7765 */  /* M10.1.085 */ /* 34085ms */
```

Comments are inside /* ... */ or start with // till end of line.

Pauses are shown before commands either in MIDI units (47;) or musical notation (536/4;).

MIDI notes consists of pairs of commands: Note on is displayed as + and Note off is displayed as -

MIDI channel are displayed in mtrk(channel number) or as [channel number] inside a track before a command.

Values are shown decimal (0-127) or hexadecimal ($00 - $7F).

Most commands begin with a keyword and have parameters e.g.

```
  beats 142.38051
  text "Contact: info@gnmidi.com"
```
  program Ensmble1

Program names (of GM instruments) can be used by number 0-127 or from following GM instrument list:

```
  0 GrandPno
  1 BritePno
  2 El.Grand
  3 HnkyTonk
  4 ElPiano1
  5 ElPiano2
  6 Harpsich
  7 Clavi.
  8 Celesta
  9 Glocken
 10 MusicBox
 11 Vibes
 12 Marimba
 13 Xylophon
 14 TubulBel
 15 Dulcimer
 16 DrawOrgn
 17 PercOrgn
 18 RockOrgn
 19 ChrcOrgn
 20 ReedOrgn
 21 Acordion
 22 Harmnica
 23 TangoAcd
 24 NylonGtr
 25 SteelGtr
 26 JazzGtr
 27 CleanGtr
 28 MuteGtr
 29 Ovrdrive
 30 Distortd
 31 Harmnics
 32 WoodBass
 33 FngrBass
 34 PickBass
 35 Fretless
 36 SlapBas1
 37 SlapBas2
 38 SynBass1
```

```
 39 SynBass2
          40 Violin
          41 Viola
          42 Cello
          43 Contra
          44 TremStrg
          45 Pizzicto
          46 Harp
          47 Timpani
          48 Ensmble1
          49 Ensmble2
          50 SynStrg1
          51 SynStrg2
          52 AahChoir
          53 OohChoir
          54 SynChoir
          55 OrchHit
          56 Trumpet
          57 Trombone
          58 Tuba
          59 MuteTrum
          60 FrenchHr
          61 BrasSect
          62 SynBras1
          63 SynBras2
          64 SprnoSax
          65 AltoSax
          66 TenorSax
          67 BariSax
          68 Oboe
          69 EnglHorn
          70 Bassoon
          71 Clarinet
          72 Piccolo
          73 Flute
          74 Recorder
          75 PanFlute
          76 Bottle
          77 Shakhchi
          78 Whistle
          79 Ocarina
          80 SquareLd
          81 SawLd
          82 CaliopLd
          83 ChiffLd
          84 CharanLd
          85 VoiceLd
          86 FifthLd
          87 Bass&Ld
          88 NewAgePd
          89 WarmPd
          90 PolySyPd
          91 ChoirPd
          92 BowedPd
          93 MetalPd
          94 HaloPd
          95 SweepPd
          96 Rain
          97 SoundTrk
          98 Crystal
          99 Atmosphr
         100 Bright
         101 Goblin
         102 Echoes
         103 SciFi
         104 Sitar
```

```
105 Banjo
        106 Shamisen
        107 Koto
        108 Kalimba
        109 Bagpipe
        110 Fiddle
        111 Shanai
        112 TnklBell
        113 Agogo
        114 StlDrum
        115 WoodBlok
        116 TaikoDrm
        117 MelodTom
        118 SynthTom
        119 RevCymbl
        120 FretNoiz
        121 BrthNoiz
        122 Seashore
        123 Tweet
        124 Telphone
        125 Helicptr
        126 Applause
        127 Gunshot
```

The names must be written exactly.

The following scheme explains the syntax of MIDI ASCII text in E-BNF (Extended Backus-Naur-Form).

Extended BNF rules:

```
symbol ::= expr ;           rule for symbol
expr can be:
expr*                       optional list of expr's
expr+                       repetition of expr's (at least 1)
[expr]                      optional expr  (0 or 1 occurrences)
expr1 expr2 ... exprN       sequence of expr1 ... exprN (in this order)
expr1|expr2|...|exprN       alternatives between expr1...exprN (choose one)
(expr)                      expr itself for grouping (e.g. ("+"|"-")*
"mthd" "("                    keywords and operators (case sensitive, use
                                        without " characters)
// text                     comment until next line
literal   ::=   characters enclosed in "..."  e.g. "Track 1"
                            (special characters can be escaped by preceding \
                            e.g. "\""  is " character itself
```

lexical symbols (in E-BNF):

```
digit     ::=   "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9" ;
hexdigit  ::=   digit|"A"|"B"|"C"|"D"|"E"|"F"|"a"|"b"|"c"|"d"|"e"|"f" ;
decnumber ::=   digit+ ;
hexnumber ::=   "0" ("x"|"X") hexdigit+
                            | "$" hexdigit+
                            ;
number    ::=   decnumber | hexnumber ;
floatnumber ::= decnumber | decnumber "." decnumber ;
notekey   ::=   "C"|"D"|"E"|"F"|"G"|"A"|"H"|"B"
                | "c"|"d"|"e"|"f"|"g"|"a"|"h"|"b" ;
                            // notekey "h" used in German language is equal to
                            // notekey "b"

notename  ::=   notekey ["#" | "is" | "b" | "es"] decnumber ;
                            // only legal notes as known in music theory are allowed
                            // suffixes "is" and "es" are used in German language
                            // "is" is equal to "#"
                            // "es" is equal to "b"
                            // exceptions rules (in German language):
                            //          As is equal to Ab
                            //          Es is equal to Eb
                            //          Use F instead of Eis
                            //  Use C instead of His
                            //          Use As instead of Aes
```

```
                        //  Use Es instead of Ees
                        //  Use H instead of Ces
                        //  Use E instead of Fes

note     ::=   notename | number ;

programname ::=
  // korg i2/i3 only:
  "Dr1"|"Dr2"|"Dr3"|"Dr4"|"Dr5"|"Dr6"|"Dr7"|"Dr8"|
  // new program names
  "GrandPno"|"BritePno"|"El.Grand"|"HnkyTonk"|"ElPiano1"|"ElPiano2"|
  "Harpsich"| "Clavi."|"Celesta"|"Glocken"|"MusicBox"|"Vibes"|"Marimba"|
  "Xylophon"|"TubulBel"|"Dulcimer"|"DrawOrgn"|"PercOrgn"|"RockOrgn"|"ChrcOrgn"|
  "ReedOrgn"|"Acordion"|"Harmnica"|"TangoAcd"|"NylonGtr"|"SteelGtr"|"JazzGtr"|
  "CleanGtr"|"MuteGtr"|"Ovrdrive"|"Distortd"|"Harmnics"|"WoodBass"|"FngrBass"|
  "PickBass"|"Fretless"|"SlapBas1"|"SlapBas2"|"SynBass1"|"SynBass2"|"Violin"|
  "Viola"|"Cello"|"Contra"|"TremStrg"|"Pizzicto"|"Harp"|"Timpani"|"Ensmble1"|
  "Ensmble2"|"SynStrg1"|"SynStrg2"|"AahChoir"|"OohChoir"|"SynChoir"|"OrchHit"|
  "Trumpet"|"Trombone"|"Tuba"|"MuteTrum"|"FrenchHr"|"BrasSect"|"SynBras1"|
  "SynBras2"|"SprnoSax"|"AltoSax"|"TenorSax"|"BariSax"|"Oboe"|"EnglHorn"|
  "Bassoon"|"Clarinet"|"Piccolo"|"Flute"|"Recorder"|"PanFlute"|"Bottle"|
  "Shakhchi"|"Whistle"|"Ocarina"|"SquareLd"|"SawLd"|"CaliopLd"|"ChiffLd"|
  "CharanLd"|"VoiceLd"|"FifthLd"|"Bass&Ld"|"NewAgePd"|"WarmPd"|"PolySyPd"|
  "ChoirPd"|"BowedPd"|"MetalPd"|"HaloPd"|"SweepPd"|"Rain"|"SoundTrk"|
  "Crystal"|"Atmosphr"|"Bright"|"Goblin"|"Echoes"|"SciFi"|"Sitar"|"Banjo"|
  "Shamisen"|"Koto"|"Kalimba"|"Bagpipe"|"Fiddle"|"Shanai"|"TnklBell"|
  "Agogo"|"StlDrum"|"WoodBlok"|"TaikoDrm"|"MelodTom"|"SynthTom"|"RevCymbl"|
  "FretNoiz"|"BrthNoiz"|"Seashore"|"Tweet"|"Telphone"|"Helicptr"|"Applause"|
  "Gunshot"|
// old general MIDI programs (GM):
  "Piano"| "BritePiano"| "HammerPiano"| "HonkeyTonk"| "NewTines"| "DigiPiano"| "Harpsicord"| "Clav"|
  "Celesta"| "Glocken"| "MusicBox"| "Vibes"| "Marimba"| "Xylophon"| "Tubular"| "Santur"|
  "FullOrgan"| "PercOrgan"| "BX-3Organ"| "ChurchPipe"| "Positive"| "Musette"| "Harmonica"| "Tango"|
  "ClassicGtr"| "A.Guitar"| "JazzGuitar"| "CleanGtr"| "MuteGuitar"| "OverDrive"| "DistGuitar"| "RockMonics"|
  "JazzBass"| "DeepBass"| "PickBass"| "FretLess"| "SlapBass1"| "SlapBass2"| "SynthBass1"| "SynthBass2"|
  "Violin"| "Viola"| "Cello"| "ContraBass"| "TremoloStr"| "Pizzicato"| "Harp"| "Timpani"|
  "Marcato"| "SlowString"| "AnalogPad"| "StringPad"| "Choir"| "DooVoice"| "Voices"| "OrchHit"|
  "Trumpet"| "Trombone"| "Tuba"| "MutedTrumpet"| "FrenchHorn"| "Brass"| "SynBrass1"| "SynBrass2"|
  "SopranoSax"| "AltoSax"| "TenorSax"| "BariSax"| "SweetOboe"| "EnglishHorn"| "BasoonOboe"| "Clarinet"|
  "Piccolo"| "Flute"| "Recorder"| "PanFlute"| "Bottle"| "Shakuhachi"|"Whistle"| "Ocarina"|
  "SquareWave"| "SawWave"| "SynCalinope"| "SynChiff"| "Charang"| "AirChorus"| "Rezzo4ths"| "Bass&Lead"|
  "Fantasia"| "WarmPad"| "PolyPad"| "GhostPad"| "BowedGlas"| "MetalPad"| "HaloPad"| "Sweep"|
  "IceRain"| "SoundTrack"| "Crystal"| "Atmosphere"| "Brightness"| "Goblin"| "EchoDrop"| "StarTheme"|
  "Sitar"| "Banjo"| "Shamisen"| "Koto"| "Kalimba"|"Scotland"|"Fiddle"| "Shanai"|
  "MetalBell"| "Agogo"| "SteelDrums"| "Woodblock"| "Taiko"| "Tom"| "SynthTom"| "RevCymbal"|
  "FretNoise"| "NoiseChiff"| "Seashore"| "Birds"| "Telephone"| "Helicopter"| "Stadium!!"| "GunShot"
  ;
```

Grammar in extended BNF

```
midifile ::= midisong
  ;

midisong ::= songoption* midihead songoption* miditrack+
  ;

songoption ::= "mute" channel+     // ignore these channels
  | "solo" channel+                // use these channels only
  ;

channel ::= number                 // only 1-16 are valid channels
  ;                              // channel 10 should be used for drums

midihead ::= "mthd" [version] [unit] "end" "mthd"
  ;

version ::= "version" number   // default version: 1
  // currently only versions 0-2 are allowed
  // version 0 = single multichannel track
  // version 1 = some singlechannel tracks playing together
  // version 2 = some multichannel tracks playing one after one
```

```
unit ::=  "unit" number              // default unit: 192
  ;

miditrack ::= "mtrk" [ "(" channel ")" ]
                        event*
                  "end" "mtrk"
                  ;

event ::= [ "[" channel "]" ] midievent
  | "velocyon" number    // default is 127
  | "velocyoff" number   // default is 0
  | duration sep         // pause: delay between events
  | "print" sep
  | "transpose" ["+"|"-"] number sep
  | sep
  | "copy" "part" literal ;
  | "part" literal
             event*
    "end" "part" literal
    ;
  | "loop" number
             event*
    "end" "loop"
    ;

sep ::= ";";

midievent ::=
    "seqnumber" number
  | "text" literal
  | "copyright" literal
  | "trackname" literal
  | "instrument" literal
  | "lyric" literal
  | "prefixchannel" channel  // following sysex or meta event is applied to this channel
  | "prefixport"   number   // following sysex or meta event is applied to this port
  | "smpteofs" number number ":" number ":" number ":" number ":" number
                          // SMPTE mode hour:minute:second:frame:fractional_frame
                          // mode 0:  24 frames/second
                          // mode 1:  25 frames/second
                          // mode 2:  30 frames/second allow dropping frames
                          // mode 3:  30 frames/second no dropping allowed
  | "tact" number "/" number number number  // tactnom / (2 ^^ tactdenom)  clicks/beat 32th/beat
  | "tempo" number  // microseconds per quarter note
  | "beats" floatnumber  // same as 60.000.000/tempo
                                  // quarter notes per minute
  | "key" literal      // literal must contain a valid key:
             // "Cmin" "Cmaj" "1bmin" "1bmaj" ... "7bmaj" ... "7#min" "7#maj"
  | "event"           // enter event bytes without change
      bytes       // no length is added
    "end" "event"
  | "metaevent" number  // metaevent nr. 0-127
      bytes             // length will be automatically added
    "end" "metaevent"    // metaevents are 0xff-codes
  | "psrmeta" "chord" literal psrbasschord  // special meta events for Yamaha PSR  chords
  | "sysevent"                  // sysex event (0xf0)
      bytes                          // length will be automatically added
    "end" "sysevent"
                                  // end sysevent code 0xf7 is appended automatically!
  | "syshex"                    // sysex event (0xf0)
      hexbytes          // length will be automatically added
    "eox"
                                  // end sysevent code 0xf7 is appended automatically!
  | "gmreset"                     // common sysex command to set GM mode on
  | "gsreset"                     // common sysex command to set GS mode on (mainly used for Roland, Yamaha)
  | "gsenter"                     // same as command gsreset
  | "gsexit"              // common sysex command to set GS mode off (mainly used for Roland, Yamaha)
  | "program" (programname | number | ("A"|"B"|"C"|"D") number
  | "control" number number
  | "hbank" number
  | "lbank" number
```

```
| "banka"
  | "bankb"
  | "bankc"
  | "bankd"
  | "bankdrum" // bank*:  korg i2/i3 only!
  | "balance" ("left" | "right" | number)
      // number is a value between 0 and 127: 0 is left and 127 is right
  | "hold" ("on" | "off" | number)
  | "reverb" number
  | "chorus" number
  | "brightness" number
  | "expression" number
  | "pitchmodulation" number
  | "wheel" number
  | "breath" number
  | "foot" number
  | "portamentotime" number
  | "portamento" number
  | "data" number
  | "volume" number
  | "sustain" number
  | "sostenuto" number
  | "softpedal" number
  | "datainc" number
  | "datadec" number
  | "highRPN" number
  | "lowRPN" number
  | "pitchbendrange" number
  | "localon"
  | "localoff"
  | "silent"
  | "allnotesoff"
  | "omnioff"
  | "omnion"
  | "monoon"
  | "polyon"
  | "songpos"  number
  | "songselect" number
  | "tunerequest"
  | "timingclock"
  | "start"
  | "continue"
  | "stop"
  | "activesensing"
  | "polyaftertouch" note number
  | "aftertouch" number
  | "pitch bend" number
  | notename duration number sep
  | "+" notename (number|"velocyon") sep
  | "-" notename (number|"velocyoff) sep
  | "+" number number sep  // note on  with velocity
  | "-" number number sep  // note off with velocity
  ;

psrbasschord ::=  // optional
   | "basschord" literal ;

duration ::=
    number             // units as defined in header
  | number "/" number  (tact units, e.g. 3/4)
  ;

program ::= number | programname ;

bytes ::= (number|literal)+ ;

hexsequence ::= (hexdigits|literal)+ ;

hexdigits ::= hexdigit+ ;
```

3.54 Guess song key and optionally set MIDI song key

[in menu Analyse]

This operation analyzes the MIDI song and suggests a song key that you can assign to the song.
It chooses the song key (number of # or b, major or minor) that will produce minimum number of #, b exceptions on a score sheet. Change the selected key if you want to set an other key.

Answering **yes** will write the song key information (META event) into the MIDI file.

Hint: A song could be printed on score sheet with any key, with a bad key choice the number of necessary exception symbols (#, b) within the measures might be high and the scores might be difficult to read.

3.55 Show original MIDI song keys

[in menu Analyse]

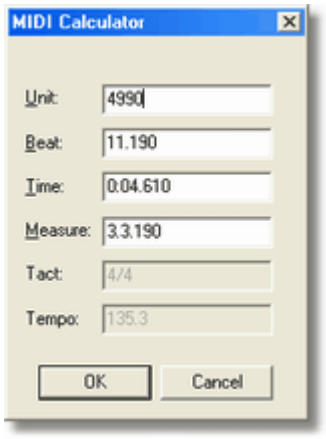
This operation analyzes the MIDI file and writes a list of used key settings into a text file that is shown with notepad text browser.

The song position (MIDI unit) is shown where the key is changed, and key name meaning is explained by the number of # or b on a score sheet.

e.g. key C Major has no #, key Am has no b,
key D Major has two #, key Bm has 2 b.
Default key is C, if it is not set within song.

```
MIDI-Unit     Song key
00000000:     C (0 #)
```

3.56 MIDI time calculator (calculate position, time, tact, tempo within a song)



[in menu Analyse]

The MIDI calculator does position calculations for current MIDI song between different position formats. Enter a position in one of the 4 different position fields (in correct format for the field) and it automatically calculates corresponding positions in the other formats and shows meter and tempo info at this position.

Unit
MIDI unit number, exact position within a MIDI file (0=start of song)

Beat
Number of quarter notes since start of song, and remaining MIDI units within the beat (starts at 1.000)

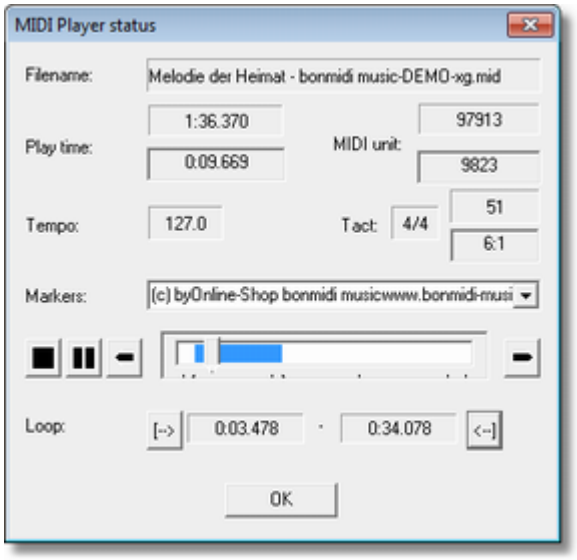
Time
Time position since start of song (minutes:seconds.milliseconds)

Measure
Bar position (measures.beat.unit) starts at 1.1.000

Tact
meter info, length of the measure at given position (nominator/denominator) e.g. 4/4, 3/4, 6/8 ..., this information can't be modified here

Tempo
tempo (bpm, beats per minute e.g. 120.0) , this information can't be modified here

3.57 Player status information



[in menu Player]

The player status dialog can be displayed while playing a MIDI song with internal player. It contains information about the playing position and has button controls to navigate the player.

Filename
path of the currently played MIDI file

Play time
current play (or pause) time position (in minutes:seconds.milliseconds). While the player is paused it is allowed to enter a new play time position into the edit field. After short pause the song plays from the position if the entered value is a valid play time.
Possible inputs are: 120 (second), 1:30 (minute:second), 3:22.350 (minute:second:millisecond)

MIDI unit
current MIDI position (in MIDI units). While the MIDI player is paused it is allowed to enter a new play unit position (a number) into the edit field. After short time the song plays from the unit position if it was a valid unit number within song.

Tempo
current MIDI tempo (in bpm)

Tact
current measure length (in nominator/denominator)

Measure:Beat
current MIDI song position (measure number, beat number) e.g. counts 23:1, 23:2, 23:3, 23:4 in a 4/4 measure. While the player is paused it is allowed to enter a new MIDI song position into the edit field. After short time the song plays from the song position if it is a valid position within the song.
Possible inputs are: 47 (measure number counting beginning from 1), 23:3 (measure number and beat number counting from 1).

Hint: In a 4/4 measure the beats within the measure are 1,2,3,4. Beat 5 belongs to next measure and therefore 23:5 is same as 24:1

Hint: 1:100 is the 100th beat within the song (independent of the measures).

Markers
the drop down box contains marker text found in the MIDI file which are sorted by time. During playing the marker directly before current playing position will be displayed. After selecting a marker from the box the player jumps to the marker position in the song. Markers usually define beginnings of sections (e.g. intro, refrain, verse) or notices to positions in the song (e.g. contra point, scalar transposition ...).

 **Stop**
stops playing the song

 **Play**
starts or continues to play current song

 **Pause**
pauses current song

 **Backward**
jumps to position 15 seconds before current position

 **Position slider**
the slider shows current song playing position relative to song duration. The duration between the position markers is 30 seconds. You can drag the thumb by clicking left mouse button on the thumb and move the mouse. The player will jump to the position where the mouse button is released.

 **Forward**
jumps to position 15 seconds after current position

Loop
Using the begin loop **[-->** and end loop **<--]** buttons a time range can be defined where the player loops.
First button click sets a position. Second button click removes the position (no loop).
The loop position times are shown near the buttons and a valid loop range is highlighted in the position slider.

Hint: when beginning position is bigger than ending position then both positions are swapped.

3.58 Insert Marker

[in menu modify]

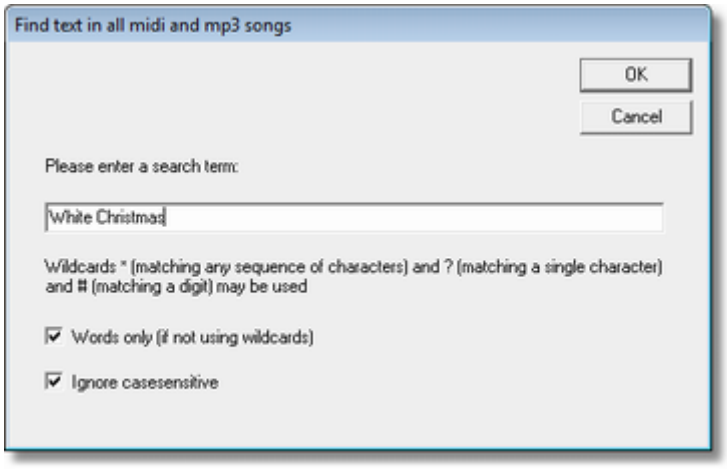
This operation is only available while a MIDI song is played and the player is currently paused. Use the MIDI Player Status dialog to move current play position to a location where you want to add a marker (e.g. intro, refrain, verse, scale transposition ...). Use the pause operation  to hold this position and then start the insert marker operation.

A dialog opens where you can enter a text that describes the mark (single line). Then use OK button to write the marker into the current active MIDI song (be sure that the correct MIDI file is active if more files are open).

The new marker will be known to the player only when you play the new result MIDI file.

Abort the dialog using cancel if no modifications are wanted.

3.59 Find text in MIDI and MP3 files



[in menu Analyse]

This operation searches text in MIDI or MP3 files in a chosen folder and all music files in the sub folders.

First search might take long time, information will be collected for future searches. Additional searches in this folder will be quicklier. The **search optimization** only works with not write protected folders. GNMIDI generates a file **gnmidtxt.fnd** in each not write protected folder.

The **matching filenames** are written into a text file that will be shown with notepad editor.

MP3 information:
the word is searched in filename (often contains artist and song title) and MP3 ID3 tag fields.

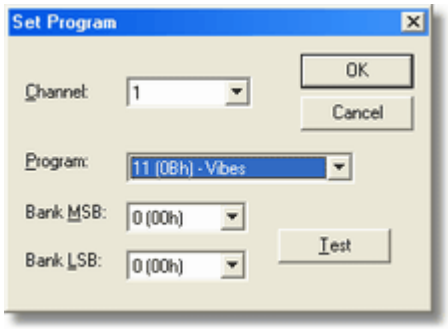
Wildcards wildcard characters can be used for matching some unknown parts
* match empty or any character sequence
? match any single character
match any single digit

e.g. w*it?e* matches text white

Words only (if not using wildcards)
the given text only matches at beginning of a word and ending of a word (it does not match in middle of a word)
when the pattern contains *, ?, # then the option is ignored
Without this option a pattern text matches also part of words e.g. ike matches also a text like

Ignore casesensitive
the pattern text can be written with lower or upper case and still can match e.g. WHITE matches white

3.60 Set programs and banks (with sound check)



[in menu Modify/Sound operations as set sound program]

This operation shows initial sound program addresses for each channel and lets assign new sound programs. The operation removes all program and bank settings of channels that were modified in this dialog and inserts the new settings before first note starts to play.

Channel
Select a channel to inspect the currently assigned sound program for this channel.

Program
Select a sound program number between 0 and 127. The GM instrument names are displayed in the list. Some device manuals count the program number from 1-128 (in this case you need to subtract 1 for entering the value here).

Bank MSB and Bank LSB
Use bank changes to access more than 128 programs on your sound device (both values are required, use 0 if the manual does not specify one of the values).

Hint: When you change a program or bank parameter the info is remembered for all 16 channels till you exit the dialog.

Hint: Your device manual contains a table of available sounds and their values. General MIDI has only 128 GM Programs and a GM standard drum kit (channel 10).

Some manuals show the parameters in **hexadecimal numbers**, here is a table to **convert the hex**

numbers to decimal numbers for 0-127:

```
00->  0  10-> 16  20-> 32  30-> 48  40-> 64  50-> 80  60-> 96  70->112
01->  1  11-> 17  21-> 33  31-> 49  41-> 65  51-> 81  61-> 97  71->113
02->  2  12-> 18  22-> 34  32-> 50  42-> 66  52-> 82  62-> 98  72->114
03->  3  13-> 19  23-> 35  33-> 51  43-> 67  53-> 83  63-> 99  73->115
04->  4  14-> 20  24-> 36  34-> 52  44-> 68  54-> 84  64->100  74->116
05->  5  15-> 21  25-> 37  35-> 53  45-> 69  55-> 85  65->101  75->117
06->  6  16-> 22  26-> 38  36-> 54  46-> 70  56-> 86  66->102  76->118
07->  7  17-> 23  27-> 39  37-> 55  47-> 71  57-> 87  67->103  77->119
08->  8  18-> 24  28-> 40  38-> 56  48-> 72  58-> 88  68->104  78->120
09->  9  19-> 25  29-> 41  39-> 57  49-> 73  59-> 89  69->105  79->121
0A-> 10  1A-> 26  2A-> 42  3A-> 58  4A-> 74  5A-> 90  6A->106  7A->122
0B-> 11  1B-> 27  2B-> 43  3B-> 59  4B-> 75  5B-> 91  6B->107  7B->123
0C-> 12  1C-> 28  2C-> 44  3C-> 60  4C-> 76  5C-> 92  6C->108  7C->124
0D-> 13  1D-> 29  2D-> 45  3D-> 61  4D-> 77  5D-> 93  6D->109  7D->125
0E-> 14  1E-> 30  2E-> 46  3E-> 62  4E-> 78  5E-> 94  6E->110  7E->126
0F-> 15  1F-> 31  2F-> 47  3F-> 63  4F-> 79  5F-> 95  6F->111  7F->127
```

Test

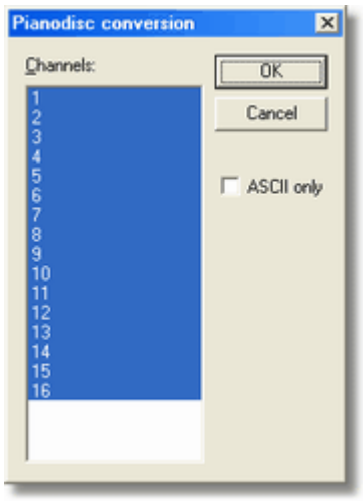
test the chosen sound before you assign it to the song press test button, this plays a small arpeggio of notes with current selected sound through internal MIDI player (any song currently playing will be stopped). This is useful if your destination device is connected to your computer (e.g. sound card, soft synthesizer, keyboard with MIDI cable) .

The test melody MIDI file can be replaced by any other MIDI file that plays notes on channel 1. Set following setting in your gnmidi.ini file if you want to use an other MIDI file:

```
[Settings]
MidiProgTest=c:\gnmidi\mytest.mid
```

Hint: Drum programs can not be tested with the test button.

3.61 Prepare MIDI file for PianoDisc



[in menu Modify/Sound operations]

PianoDisc https://www.pianodisc.com modules let a piano play the keys self ("ghost player"). MIDI files control the playing of a piano song.

This operation **converts the current MIDI song to format 0** and **assigns a special piano sound** to

the selected channels that forces the PianoDisc to play the piano keys.

Hint: The song should only play at channel 1 (use map channels or mute channels).

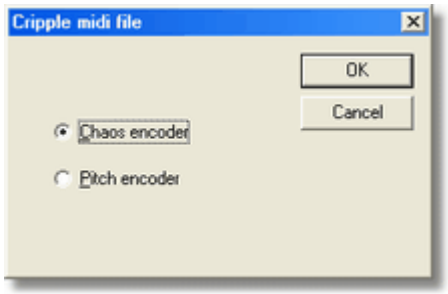
Channels
select one or more channels (default all) where the piano sound will be assigned

ASCII only
it seems that some older PianoDisc versions don't load MIDI files that contain international characters. This option replaces non-ASCII characters.

Sound address
this operation assigns sound address Program 0 (piano) MSB=5 LSB=87 in selected channels by default.
This address can be adjusted in file gnmidi.ini using following setting line:
[Settings]
Pianodisc=0 5 87

You can find the address numbers in your device manual.

3.62 Cripple notes



[in menu Modify/Note operations]

This operation encrypts the notes of the MIDI song. This is useful for making MIDI file unusable for printing or editing. The operation can't be reversed.

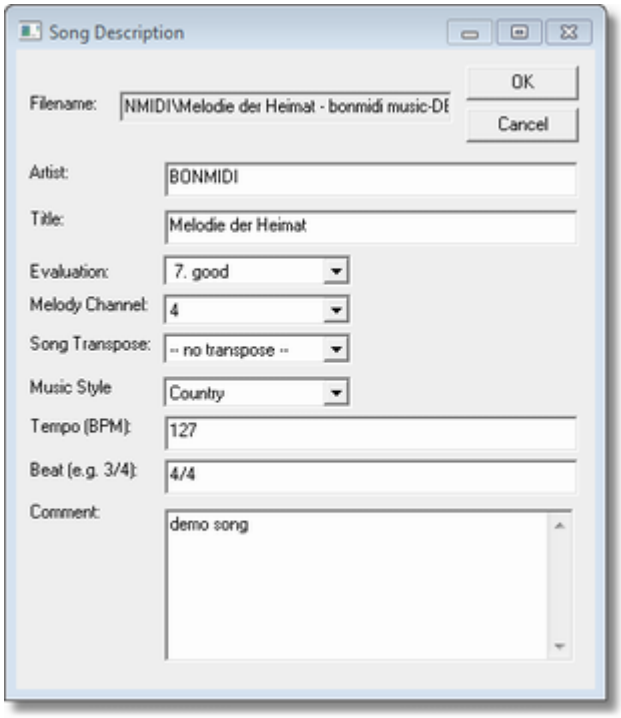
Two different methods are available:

Chaos Encoder
creates a little chaos in the song, the commands are moved into random tracks which causes bad printings but still plays correctly.

Pitch Encoder
changes notes by random half tones and uses pitch bending to put them back to pitch.

Hint: Check carefully if the song plays identically after cripple operation, since the operation applies changes to the notes and pitch bending.

3.63 Song description



This operation displays or adds additional info to the current MIDI or MP3 file.
The description to the file will be stored outside of the MIDI/MP3 file.
Some of this information will be displayed in the MIDI document window. Some of the fields can be searched. Some field values are used by the entertainment player.

Filename
Path and name of the song file

Artist
composer or singer of the song

Title
song title

Evaluation
Rate the song by one of the attributes in the list. Rating values are considered by Entertainment player.

Melody Channels
if you know which channels the melody notes then you can set this info here. 1-16 or 0 if you don't know or don't need it. The information is required by operation Mute Melody and MIDI karaoke score lines display.
A list of channel numbers can be specified with comma separator. Some operations need a single channel number. Then the channel number with smallest value is used from the list.

Hint:
Melody channel is not used for MP3 files.

Song Transpose

enter a transpose value (-12 .. +12) e.g. if you use a score sheet with the song notes in other key. Operation Mute Melody transposes the song by given value. -- no transpose -- if transposing is not necessary or not wanted.

Hint:
MP3 songs are not transposed before playing.

T**empo (BPM)**
a beats/minute value between 1 and 255.

Hint: Tempo is used by metronome bar for mp3 playing when metronome is visible and option display mp3 tempo and beat is on

Beat (e.g. 3/4)
number of beats counting, for 3/4: 1,2,3,1,2,3...

Hint: Beat is used by metronome bar for mp3 playing when metronome is visible and option display mp3 tempo and beat is on

Music Style
select a music style category from the list of categories for a song. The list of styles is read from file **gnstyles.ini** in your gnmidi directory, it will be created once you start GNMIDI. You can add categories self to this text file or choose unknown for all that have a category that is not existing in the list.

Comment
enter own commentary text to the song

Hint: GNMIDI writes this information into files **gnmidi.dsc** in the directory of the music file by default. With following setting in gnmidi.ini settings file in documents folder you can let write them into an other folder (you can choose the name and location of the folder self):

```
[Settings]
datapath=c:\gnmididata
```

3.64 Adjust volume to common level before playing midi song (optional)

This MIDI player setting Common volume can be turned on or off by checking or unchecking the menu item.
Adjusting volume level is done by changing volume and expression controllers percentually up or down so that the song maximum volume level is close to common volume level. It does not change note velocities because that could influence the sound.

If the songs are generally too loud or too silent after adjusting to common level then you should adjust your speaker volume or keyboard main volume slider.
In some cases it is not possible to reach a level close to common level, e.g. if one channel already has maximum level then it can't be increased further.

Default common volume level is 1000000, this value is written into gnmidi.ini file:

```
[Settings]
MidiAdjustVolume=1
MidiCommonVolume=1000000
```

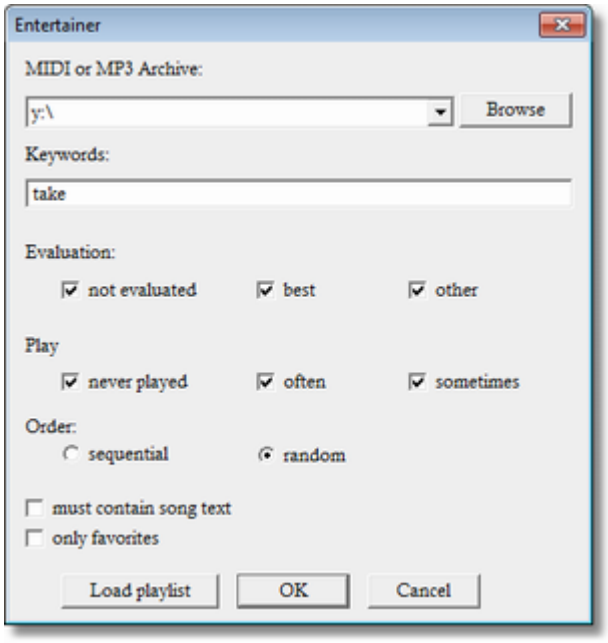
If you want change this value then use only values between 300000 and 1700000, values lower or higher lead to extreme use of volume changes which might
cause bad arrangement. If the setting is checked then it is turned on and starting MIDI player (play or play extern) will produce a temporary MIDI file for playing that has adjusted volume. In some cases volume is already at limit and can't be adjusted to the wanted level (e.g. if expression controls or note velocities prevent from getting higher volume).

3.65 Set all MIDI to common volume level

This batch operation applies same function all MIDI files in specified MIDI folder that option common volume does to current MIDI song.
Using GNMIDI Light license batch operations are not available.

This can also be done **mp3 files** (normalize volume) if FFmpeg package is installed.

3.66 Entertainment player



[in menu Player]

The Entertainment music player chooses MIDI and MP3 files from your song archive and plays them (e.g. in random order) one after one.

The operation offers search criteria's to restrict the entertainment to certain songs from your archive.

MIDI/MP3 Archive
select your previous used song archive folder from the drop down list box or use **browse button** to add a new song folder. Only MIDI or MP3 songs from this folder or its sub folders are considered for entertainment.

Keywords
enter one or more simple words (separated by spaces), all keywords must appear somewhere in the

MIDI/MP3 file text if the keyword field is not empty. The words might match filename, author, title, comment, lyrics, tracknames, markers, copyright, ...
Wildcards * and ? are allowed as search patterns (reduces search speed).

Evaluation:
each song can be rated, these values are considered here. Select between

- not evaluated — considers songs that have no rating
- best — considers songs that have good rating
- other — considers songs that have bad rating

Play Frequency
every time when a song is played with the Entertainment player, a frequency counter will be increased for a song. Select between

- never played — considers songs that were never played with Entertainment player
- often — considers songs that were often played with Entertainment player
- sometimes (but at least once) — considers songs that were not often played with Entertainment player

Order
the Entertainment player supports two playing orders

- sequential criteria's — plays the songs in the order of a play list or as they match the search
- random order — plays the songs from the play list or song archive folder in random

Must contain song text
this option can be checked to load only songs that contain song text

Only favorites
if checked then only play songs that are in favorites list

Load Playlist
instead of considering whole song archive folder you can load the list of filenames from a text file. Each line of the play list file should contain one filename (including folder path) . The play list files should have file extension .lst . Filenames that are not found (e.g. because of wrong spelling) are ignored. When a play list is loaded then the search fields are disabled, all valid MIDI/MP3 files in the play list are considered by the Entertainment player.

e.g. winner.lst

```
c:\MIDI\abba\thewinnertakesitall.mid
c:\MIDI\queen\wearethechampions.mid
c:\MP3\hot chocolate\everyone 's a winner.mp3
```

OK
with OK the entertainment player starts to search for matching songs in your song file archive, depending on the number of songs in the archive it can take some time. The Entertainment player starts to play a song (with internal player) when at least one matching song is found and it will continue to search for more songs in background while playing a song.

The dialog hides automatically when first matching song is found.
When a song finishes to play or when you stop the internal player manually, the next matching song in selected order will be started automatically soon.

 **Stop entertainment player**
Click on the entertainment player symbol or press Ctrl-A to stop the Entertainment playing.

Hint: While a dialog is open (e.g. status, description...) the entertainment player does not continue

automatically with next song. Close the dialog to continue.

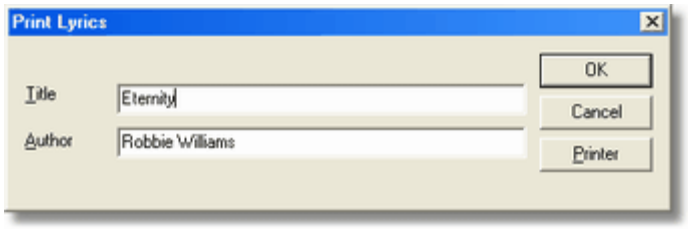
Hint: Entertainment player uses internal player which requires working installation of MIDI/MP3 MCI drivers.

3.67 Delete duplicate notes

[in menu Modify/Note operations]

This operation removes notes that are duplicated in a MIDI song. The duplicate notes must be at **same MIDI position** (or less than 3 MIDI
units away from this position) and must have s**ame channel and note number**.
If no duplicate notes are found then no result is generated and a message is displayed is status bar.

3.68 Print Lyrics



[in menu File]

Karaoke songs usually contain song text inside the MIDI file. This operation prints the song text and song title and song author.

Title
enter the song title

Author
enter the song author or composer

Printer
choose printer and

printer settings.

3.69 Pause/Continue MIDI Player commands

[in menu Player]

While a song is played by internal MIDI player you can use this command to pause the song. Use it again to continue to play the song.

In player status dialog a button exists for pause command. After pressing pause the symbol changes to a play command (continue)

3.70 Backward/Forward MIDI Player commands

[in menu Player]

While a song is played by internal MIDI player this operations are available.

Backward
continues to play 15 seconds before current song position

Forward
continues to play 15 seconds later after current song position

Both commands are available in player status dialog.

With following gnmidi.ini setting the skip time could be changed
```
[Settings]
SkipSeconds=15
```

3.71 Mute Melody

[in menu Modify/Volume operations]

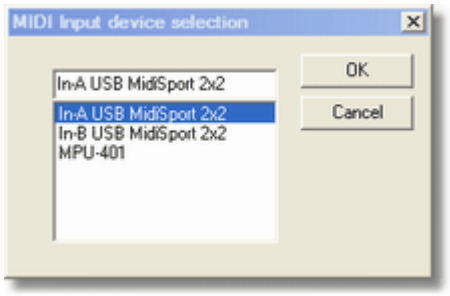
This operation removes the notes from melody channel. That is useful for karaoke purpose or to play the melody self on a MIDI keyboard along to the remaining song accompaniment.

First you need to define the MIDI melody channel of the song once in the MIDI descriptions of the file. This operation automatically starts the description dialog if the melody channel is not set.

The description dialog offers also to define a transpose value for the MIDI file. This is useful when your score sheet contains the notes in other song key and the notes (except drums) will be automatically transposed to same song key when applying this operation.

This operation can be tried also for a **Mp3 file** if FFmpeg package is installed.
In fact it only removes audio signals at center stereo pan assuming that many songs use vocals in stereo center and other instruments at left or right pan.
Hint: This surely does not work for all songs and the quality of resulting audio file might be reduced. Currently there is no better approach for removing vocals available in FFmpeg (and probably other tools).

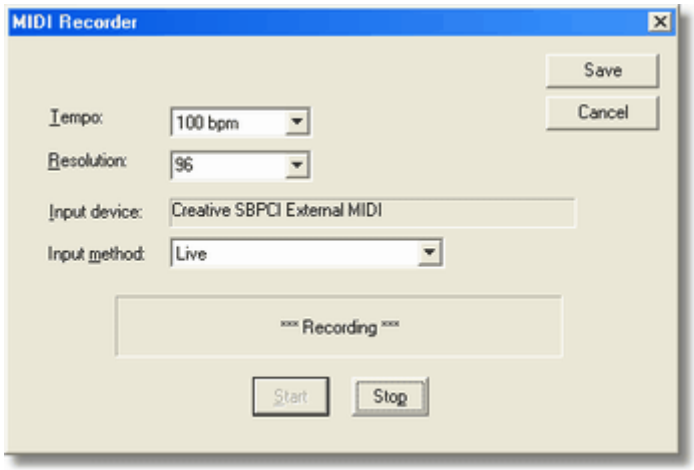
3.72 Select a Midi Input



[in menu Settings]

Select a MIDI input device from the given list of input devices that is used for recording MIDI song or recording sysex dump data.

3.73 MIDI Recorder



[in menu Player]

This operation records MIDI data from an external MIDI device through MIDI cable and stores it as a standard MIDI file. Demo program version only allows to record 20 seconds and then automatically stops recording.

Tempo
select song tempo (bpm=quarter notes/minute). The resulting MIDI file will have this constant tempo when recording using input method Live (independent which tempo you are playing).

Resolution
select song resolution (MIDI units per quarter note) for the resulting MIDI song.

Input device
select the input device in menu Settings. This field shows the current selected device name.

Input method

- Live
  real time (milliseconds) is used to record. The song will have constant tempo. This works even in MIDI modes that don't send MIDI clock signals.
- Synchronized to MIDI clock
  device must send 24 MIDI clock commands (F8) per beat. Tempo changes are guessed from speed of the clock commands.
- Delayed
  uses time stamps (milliseconds) from the input device. Tempo will be constant even if parts play with different speed..
  useful for devices that can't produce MIDI data in real time (e.g. analyzing tools) and send them delayed. This method works well with e.g. Autoscore pitch-to-MIDI software.

MIDI data receiving indicator ***
While ***** Recording ***** at the right side a **star * will blink** which indicates that MIDI clock or active sense commands are received, which are important because that means that MIDI connection works.

Start
Start recording a song from MIDI cable input

Stop
Stop the MIDI recording. A dialog will tell if MIDI data was received that could be saved.

Save
Generate a MIDI song from the recorded input data. This will open a temporary MIDI document window. Don't forget to save this document to a MIDI file.

Important:
It is necessary that the Windows MIDI device driver is correctly installed and working, that data can be recorded successfully. Some keyboards or synthesizers send MIDI data only in certain mode (e.g. song mode) or need keyboard settings to enable sending of MIDI data. Some keyboards you need to force to send initialization settings (sounds, volume...) through MIDI cable, by pressing e.g. reset button, changing mode or similar.

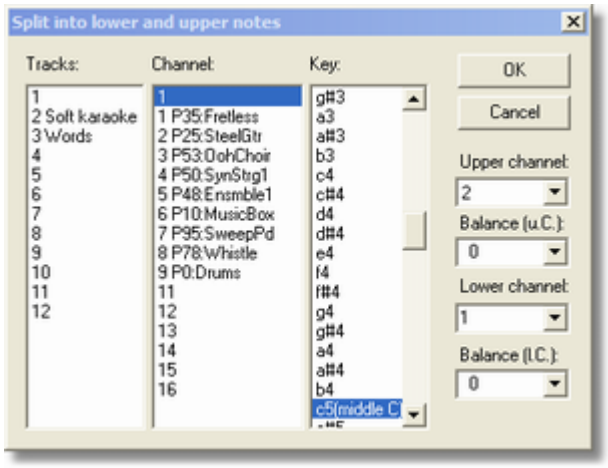
Hint: some keyboards do not automatically send MIDI clock commands through cable and need some setting on the keyboard to enable the sending.

Hint: many keyboards do not send demo song through midi cable and some keyboards not even send any recorded song as midi (not very nice from keyboard developer).

Hint: most keyboards/pianos do not send initialisation when pressing play button (they have loaded the initialisation already before playing). It may help to press STOP button directly before play button (while GNMIDI records the input).

Hint: use the operation clean MIDI after recording was successfully. This would delete not recorded channels and ignore unnecessary commands like Grandpiano sound program setting at beginning (all track show Grand piano as sound) when later the correct used sound program is sent.

3.74 Split notes into lower and upper half at splitpoint (left and right hand)



[in menu Modify/Note operations]

This command splits a channel at a given splitting point note into lower track (left hand) and upper track (right hand). You can optionally assign new MIDI channels to both parts so that both parts can play with different sounds.

Tracks
optionally select a track number only if the channel number is used in two different tracks and you want to split only one of them.

Channel
select a channel number that contains notes for left and right hand

Key
a note number where to split the selected channel into two parts. **Middle piano C is called C5** in this list. The key note and higher notes belong to the upper part (right hand), the notes that are lower than key note belong to the lower part (left hand).

Upper channel
channel number of upper part (right hand)

Balance (u.C.)
Balance -64...0...63 of the upper part (right Hand), 0 is center, -64 is full left, 63 is full right

Lower channel
channel number of lower part (left hand)

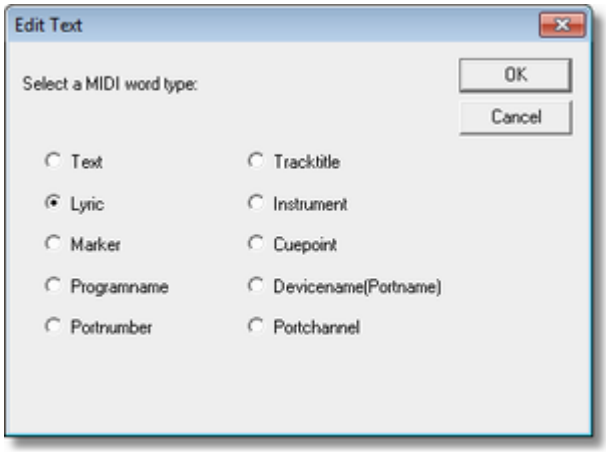
Balance (l.C.)
Balance -64...0...63 of the lower part (left Hand), 0 is center, -64 is full left, 63 is full right

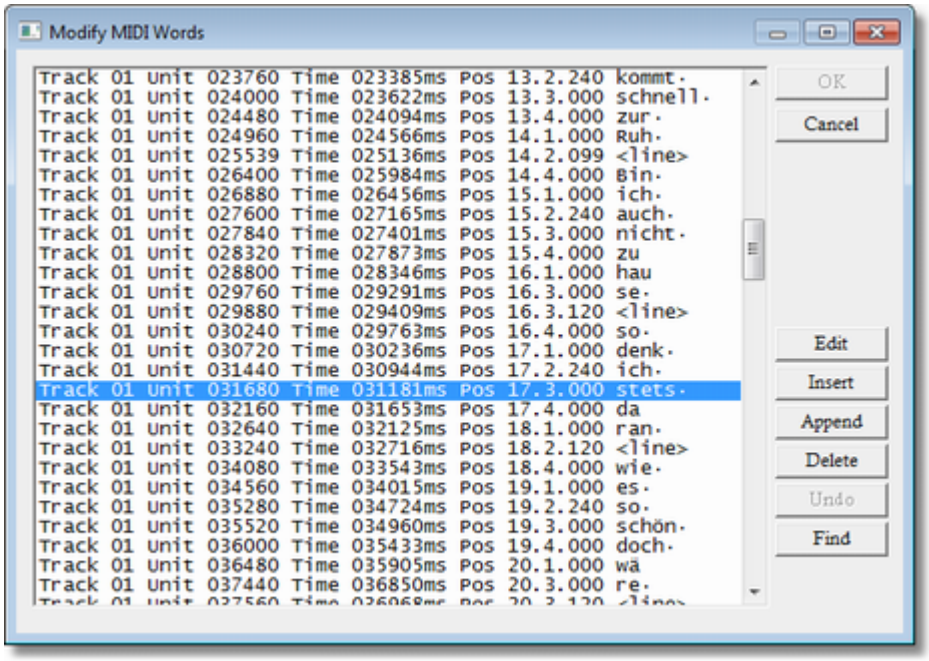
Hint: Middle C (piano-C) is named here as C5. Elder GNMIDI versions (till 2.49) called this **note C4**. If you prefer to use old octave numbering then you could add following setting into GNMIDI.INI:

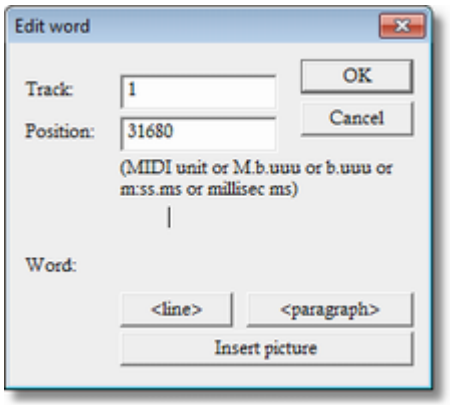
```
[Settings]
MidiMiddleOctave=4
```

Hint: Balance (also called Pan/panning) defines relation between left and right stereo output.

3.75 Edit Text







[in menu Modify]

This command is used to modify text commands and their position.

Select a MIDI word type in first dialog and continue with OK

- Text can be karaoke song text (.kar) or other text
- Lyric contains song text
- Marker can be word markers for song positions or additional text
- Tracktitle title of track, title of first track is usually used for song title
- Instrument name of used instruments or sounds
- Cuepoint name or description of a song position, e.g. refrain
- Programname a description of the used sound
- Devicename(Portname) name of a device that plays the part (usually when song is played by multiple devices)
- Portnumber META prefix port number (1-256) to assign the part to a device that has this portnumber
- Portchannel META prefix port channel (1-16) to assign a channel number to MIDI commands that have no channel (e.g. sysex).

Hint: MIDI information window contains at end of the information (you probably need to scroll the window down) contains all text of different types.

List box content

each list item shows

- Track number track that contains this text command
- Position MIDI unit, time in milliseconds, measure position (measurenr.beatnr.tick)
- Word text stored by this text command

```
Track 01 Unit 031680 Time 031181ms Pos 17.3.000 stets
```

Hint: Measure number starts with 1
Hint: Beat number starts with 1
Hint: beginning of song is 0 or 1.000 or 1.1.000
Hint: use the scroll bar to find more lines with text commands

Edit

modify the text (syllable or word or sentence) or the MIDI unit position. Use <line> and <paragraph> keywords to specify a line break or paragraph break. In .kar files the lines and paragraphs must be formatted according to .kar rules.

Entering Position

can be entered as MIDI unit (e.g. 31680) or time in milliseconds (e.g. 31181ms) or measure position (measurenr.beatnr.tick) or time (h:mm:ss.ms or mm:ss.ms)

Insert

create a new text item before current selected text item, or modify the suggested MIDI unit position to insert the item at certain MIDI position.

Append

create a new text item behind last text item, or at given MIDI unit position.

Delete

remove current selected text item

Undo

undo previous change

Find

asks for entering a word or a phrase. The text is searched in the list beginning behind previous match or from beginning.
Syllables are connected (ignoring hyphens -) and all whitespace and comma, dots etc. are replaced against a single space.
The first list entry that contains the matching phrase is selected. Use Find and <enter> to search next matching position.

Hint:
This operation does not support editing of Yamaha XF, Roland sysex video lyrics, Farfisa sysex lyrics.

Hint:
For entering and synchronizing **song text** you should use Karaoke editor and Synchronization editor.

Hint:
A text item may contain a syllable, a word or a text line.

Hint:
Track number (1-255) must reference an existing track in MIDI file, otherwise the text will not be inserted.

<line> set word to line break

<paragraph> set word to paragraph break

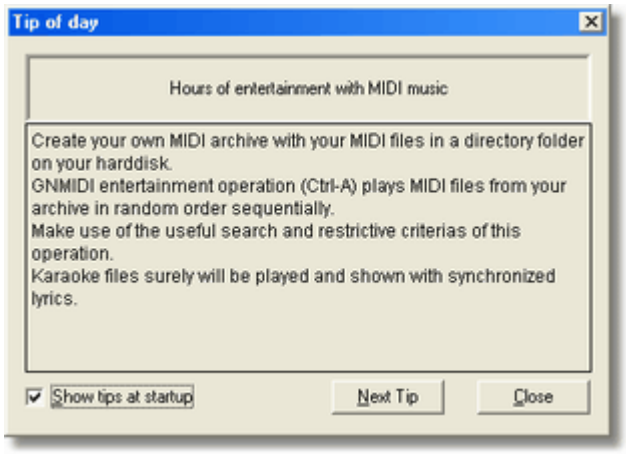
Insert picture
choose a picture from your harddisk **(*.gif, *.jpg, *.png, *.bmp)**. This inserts a **HTML tag <img src="imagepath">** into your text field. The existing image will be **displayed in the karaoke view** and printing.
Reserve a new text field if you want to assign a different synchronisation time to this picture. Use <line> that the picture is on its own line.

Hint: this can be used to insert pictures with score sheets and synchronisation helps to scroll automatically according to the song timing.

Hint: When modifying text of a **.kar file the <line>** button inserts a / before the word and the <**paragraph**> button inserts a \ before the word.

Hint: **a .kar file** uses text type TEXT and contains a word beginning with @K

3.76 Tip of the day



[in menu Help]

This dialog shows short articles about GNMIDI operations or interesting features.
At startup of GNMIDI a tip of the day might be shown. It can be activated in help menu.

Show tips at startup
If it should not start automatically at each GNMIDI session, you can deactivate this feature in the tip of the day dialog by unchecking the option.

Next Tip
show next article.

Close
exit this dialog (also <Esc> key closes dialog).

3.77 MIDI melody to mobile phone ring tones (RTTTL)

[in menu Convert]

Some mobile phones can be loaded with RTTTL ring tones. This function converts a MIDI melody channel to ring tone format.

Melody channel:
Choose the channel that should be converted. The channel must contain notes (should be mono

phon) that the conversion can be done successfully.

Result is a text file that contains the melody notes in RTTTL format.

Hint: the melody notes should not overlap because RTTTL ring tones are mono phon only.

Hint: Some mobile phones might limit the length of the RTTTL, so choose a small part of the song.

E.g. a small part of a song converted to RTTTL:

```
Eternity:d=4,o=5,b=78:16p,16d.,32p,32f,32p,8f.,8d,32f,32p,8f.,a.,g.,2p,16g.,32p,32a#
,32p,8a#.,8g,8a#,8g,16p,c.,2p,p,16p,16d.,32p,16p,8f.,8d,8f,16d.,32p,a.

The result text also offers
keystrokes for entering the melody into a Nokia 3310/3300 cell phone.
```

3.78 RTTTL to MIDI song

[in menu Convert]

This operation converts an RTTTL file loaded into a document window into a MIDI file.

The MIDI song will contain **mono phone melody**, program **Vibes** is assigned to the MIDI channel 1.

3.79 MIDI to parsons code conversion

[in menu Convert]

Parsons code describes the contour changes of a mono phon melody by simple character symbols. With parsons code it is possible to identify melody by inexact tone height changes, it does not require a precise interpretation of notes, duration and transposition.

The result will be written into a new text edit window (Notepad). Longer melodies might create long parsons code.
Longer pauses between melody notes will break the parsons code and a new parsons code will start with * in a new line.

Melody channel:
Choose a melody channel that contains the melody notes, which should be converted. This channel must contains melody notes (and should be mono phon, because Parsons code only supports mono phon melodies), that it can be converted to parsons code.
Short note overlappings will be corrected automatically. If the channel contains long note overlappings this could infect the results unwanted.

Parsons code format:

Parsons code of a melody usually begins with a * symbol, that is the placeholder for any start tone height.
Following tone changes are allowed and will be described by following characters:

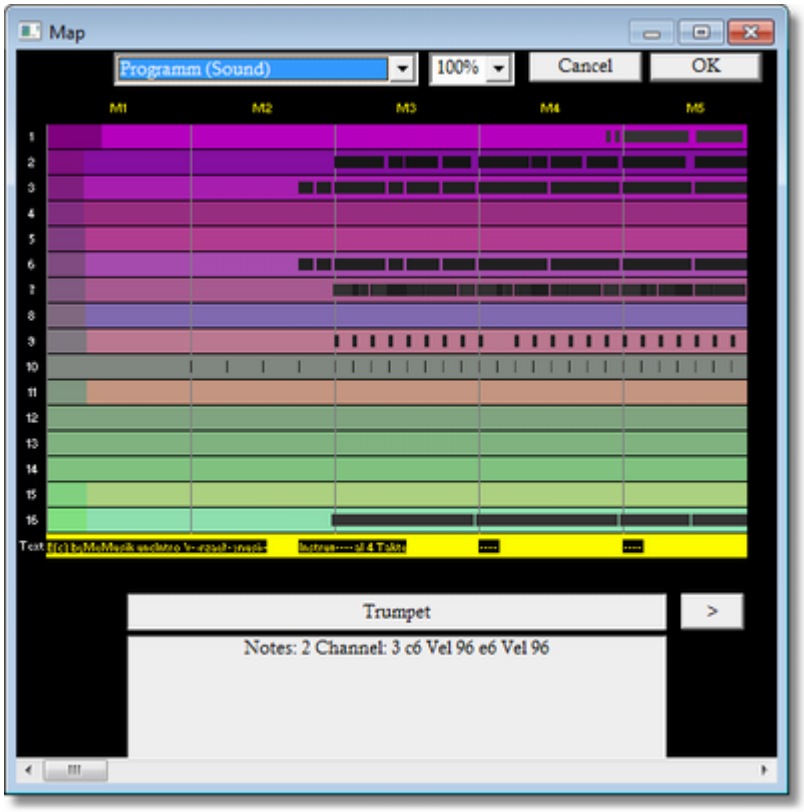
D (Down) tone height will be lower relative to the previous tone
U (Up) tone height will be higher relative to the previous tone
R (Repeat) last tone height will be repeated

Example: *DUDUDUDDDUUUDUUU
that is the Parsons code of main melody Beethoven "Per Elise" which was produced from the notes
E D# E D# E B D C A, C E A B, E G# B C.

https://www.musipedia.org provides a search engine for melody searching using parsons code. Above example was found in classic range at first place. When searching in all music ranges "Per Elise" was found at 3rd place, there are other melodies (z.B. Rolling Stones) that produce similar parsons code. It does not mean that the melodies are identical.

A conversion from Parsons code back to MIDI can not be done unique, because Parsons code does not contain notes or durations.

3.80 MIDI settings landscape view



[in menu Analyse]

The MIDI settings map displays the parameter values used within the MIDI song. Loading this info can take some time.

The coloured table has 16 rows for channels and one extra row for song lyrics.
The columns are divided into measures by grey vertical lines. The distance
is depending on the current zoom state.
The smaller black blocks inside the rows are the positions were notes are playing.
The colour is brighter if the notes are not pressed so hard (note velocity value).
The row background colours are different for each row. The colour intensity increases with the control value (e.g. volume 0-127).

Move mouse cursor over coloured areas to show the value below in the grey info field and the notes or META text at cursor position in the larger box.
Use the scroll bar arrows to show the parts that are currently outside of screen.

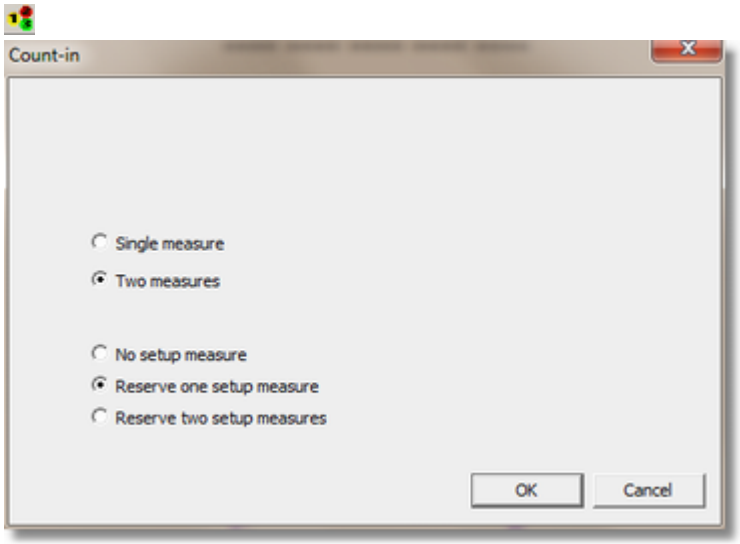
Parameters:

- Program (Sound)
- Tempo
- Volume
- Expression
- Balance
- Chorus
- Reverb

Zoom:
choose a zoom percentage value to check out a smaller range of MIDI units in more detail or a larger range of MIDI units in less detail.

Play (>)
The grey button on right bottom plays song beginning at current map position.

3.81 Count-In 1,2,3,4



[in menu Modify/Tempo operations as count-in tempo]

This operation adds some drum notes to count in the beats of initial song tempo.
You may choose to insert one measure for counting 1-2-3-4 or two measures for counting 1---3-- 1-2-3-4.
You can decide if 0, 1 or 2 first bars are reserved for MIDI initialization (setup measures) so that they are not used for count-in and insert later directly before first song note.

If the song starts with **upbeat** then the count-in is done also inside the bar that contains the beginning song notes (e.g. song starts later in a bar at ---B S S S then two measures count-in might be counted as 1-3- 123B S S S).
Hint: if song starts at 1.2.000 (upbeat) then only one count-in note might be inserted at 1.1.000 when choosing one measure count-in.

Hint: pauses will be inserted when not enough space is already available for the new count-in drum notes.

Hint: It is important that the meter information of the MIDI file is correct, so that the beginning of the first measure is at start of song (MIDI unit 0). Wrong initial measure position will result in a wrong count-in.

Hint: if there are already Count-In notes at beginning of channel 10 then no count-in notes will be inserted

Hint: a midcntin.ini settings text file could be created in your personal documents folder to define chosen drum notes, velocities options.

```
[Settings]
measures=2
drumchannel=10
drumnotefat=42
velocityfat=127
drumnotelow=44
velocitylow=100
addmarkerstars=no
```

3.82 Remove Count-in notes

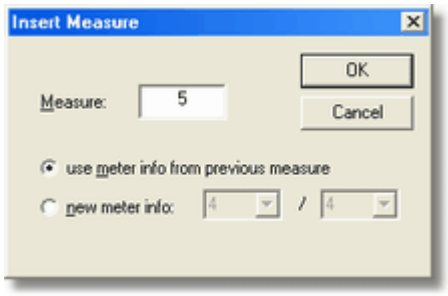
[in menu Modify/Tempo operations]

count-in notes will be identified if:

- at least 3 identical drum notes on channel 10 are played in similar distances
- the first other drum note or an other note on an other channel starts to play later
- all count-in notes must be played monophon sequentially

The this operation is available in menu modify/Tempo operations and removes these notes. Pauses are not removed.

3.83 Insert empty measure



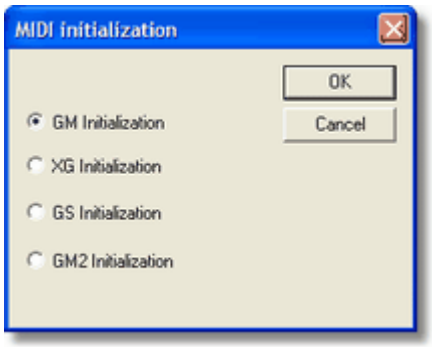
[in menu Modify]

This operation inserts a new empty measure.

Measure
Measure number of the new measure, existing measures behind this number will be moved back. First measure has number 1.

Measure length
- use meter info from previous measure — the new measure will have the same length as the measure before
- new meter info (nominator / denominator) — the new measure will get a new length (e.g. 4/4, 3/4, 6/8 ...)

3.84 Initialize GM/GS/XG/GM2 mode



[in menu Convert]

This operation adds an initialization MIDI command (**MIDI RESET**) to beginning current MIDI song.

For GM initialization it adds GM MIDI Reset Sysex and **removes GM incompatible MIDI commands** (e.g. sound bank references, sysex commands).

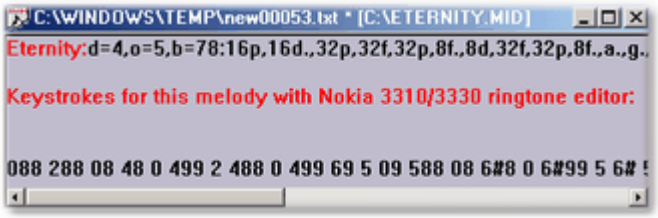
3.85 Optional MIDI compression during save operation

[in menu Settings]

Most MIDI players, keyboards, software, sound cards support this kind of MIDI compression.
MIDI Compression does not change the musical content, it still produces valid standard MIDI files, but reduces MIDI file size (up to 15% smaller).
Most files that are found in Internet or sold by music companies are compressed.

Usually this option should be checked (ON), since it only uses capabilities that MIDI standard suggests.
Turn this operation off when your MIDI device does not accept format 0 MIDI files that are compressed.
Some older Yamaha keyboards seem not to support compression and reject valid MIDI format 0 files at loading.

3.86 Keystrokes for Nokia 3310/3330 mobile phone tone editor



Nokia phone model 3330 includes a ring tone editor where a new ring tone melody can be edited by pressing the phone keys.
MIDI to RTTTL ring tone conversion creates this info and displays the keystrokes in the window.

Start the tone editor on your Nokia 3310 or 3330 for entering a new melody. Be sure that the current octave and note length are initialized.

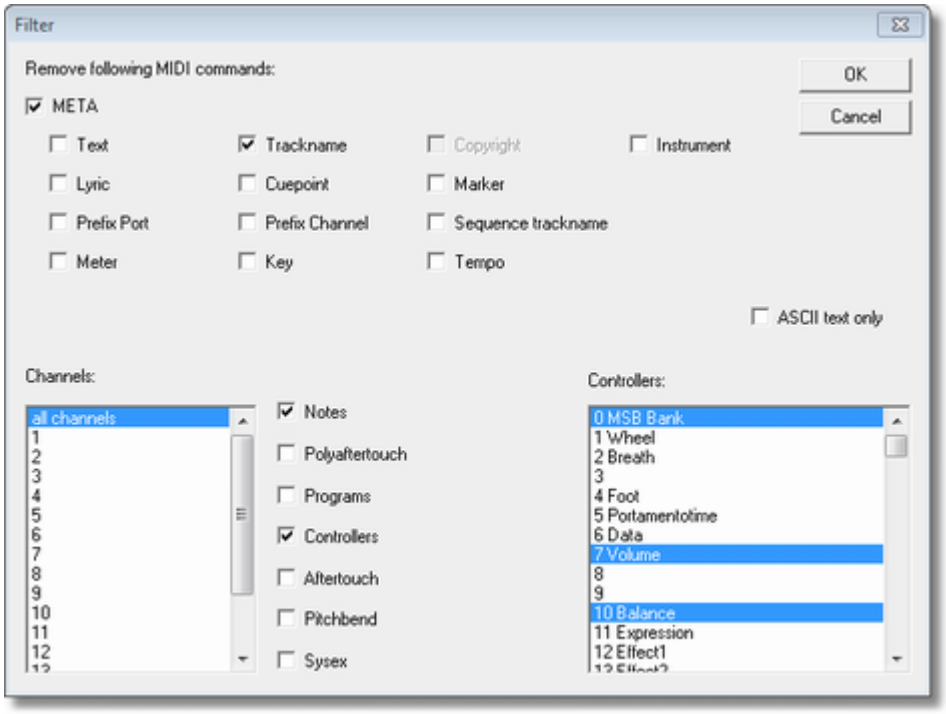
```
enter 1  that should show 4c1 on display    (if not then use keys 8,9,*)
enter <c> to remove the 4c1 from display
```

Hint: Enter the digits, *,# exactly as displayed in the document window (don't miss one, each is important to create correct melody).

Hint: The melody input on the phone might be limited (e.g. 50 notes).

Hint: It can't be guaranteed that the keystrokes are working with all mobile phones of these types (version or country depending differences can cause it to fail).

3.87 MIDI command filter



[in menu Modify/Controller operations as MIDI filter commands]

This operation removes MIDI commands (also called MIDI events) of chosen types. It can also change international text characters contained in text commands into ASCII text characters.

META commands

- Text
- Copyright (this is disabled, the operation does not allow to remove existing copyright information)
- Trackname
- Instrument
- Lyric
- Marker
- Cue point
- Prefix Port
- Prefix Channel
- Sequence trackname (trackname only in first track)
- Meter (bar length info)
- Key (song key)
- Tempo

Hint: Turn on META check box that you can select META command types

Sysex commands

remove all sysex commands when checked (check META option that this check box can be used)

Channel depending commands

- Notes

- Aftertouch
- Controller changes — select controller command numbers in right side list box
- Program changes
- Polyaftertouch
- Pitchbend

Channels
select one or more channel numbers, only those channel depending commands that match a selected channel will be removed

Controller change commands
Check box Controller changes must be selected first, then select one or more controller numbers that will be removed for selected channel numbers

ASCII text only
all non-ASCII text characters (e.g. international characters like è, ü, ß ...) are replaced against ASCII characters. This is useful if a MIDI device can't display international characters.

3.87.1 listbox multiextended selections

Some listboxes in GNMIDI support **multiextended selecting** to comfortable select one or more list items.

Select only a single list item
click with left mouse button on a list item and release it will remove all other selections and select only this list item
with arrow keys up and down the selection could be moved to previous or next list item

Select a range of adjacent list items
start by click with left mouse button on first list item and drag the mouse up or down while holding the mouse button clicked will remove all other selections and select a range of adjacent list items.
Release the mouse button when you are ready

Extend or reduce the current selections by single list item
hold the CTRL key and click on a list item. The item will be added or removed from the selection depending if the item was selected before

Extend or reduce the current selections by a a range of list items
hold the CTRL key and click on first list item and drag the mouse up or down while holding the mouse button clicked. Unselected items will become selected and selected items will become unselected.
Release the mouse button when you are ready

Replace selection to a range of adjacent items
hold the SHIFT key and click on a list item. That will select the items from the most recently selected item till the clicked item.
All other items become unselected.
Use the arrow up/down keys or the page up/down keys to select more items while holding the SHIFT key.

Select all items in the listbox
First select the first list box item. Hold the SHIFT key and use END key to select all items till last row

Select no item in the list box
Click into any list box item to select only this item. Then use CTRL and click into this listbox item to

unselect this item.

Hint: Few listboxes in GNMIDI use the simple selection mode. With that boxes you can only click and select or unselect the clicked entries.

3.88 Seek long pauses in song

[in menu Analyse]

The operation searches for positions within song that don't play a note for at least 5 seconds. The result is written to a text file and displayed by notepad editor. This operation useful when you record several songs at once with pauses between and then want to know where each part starts.

```
c:\mysong.mid:
0:04:24.230  large pause found: 0:00:17.000
0:08:53.730  ** end song
```

Time position and pause durations are displayed in format h:mm:ss.ms (hours, minutes, seconds, milliseconds).

The 5 seconds period can be changed with following setting in gnmidi.ini file:

```
[Settings]
LongPauseSeconds=7
```

3.89 Seek parts with notes

[in menu Analyse]

The operation searches groups of notes in all channels that are splitted into parts by longer pauses (minimum 5 seconds).
The result is opened in a notepad editor and contains for each channel that contains notes a list of times where notes play:
  starttime-endtime (duration)
The time format is minutes:seconds:milliseconds.

example for an output:

```
channel 1:
       0:01.875-3:15.706 (3:13.831)

channel 2:
       0:01.250-3:15.684 (3:14.434)

channel 3:
       0:00.953-0:07.646 (0:06.693)
       1:37.812-1:38.432 (0:00.620)
       1:44.218-2:20.312 (0:36.094)

channel 4:
       0:07.812-1:37.421 (1:29.609)
       2:22.812-3:15.263 (0:52.451)

channel 5:
       2:08.437-2:22.500 (0:14.063)
       2:53.437-3:15.000 (0:21.563)

channel 6:
       0:08.437-3:15.703 (3:07.266)

channel 7:
       0:08.437-3:15.625 (3:07.188)

channel 8:
       0:02.187-3:15.684 (3:13.497)

channel 9:
       0:08.437-0:12.884 (0:04.447)
       0:19.687-0:57.884 (0:38.197)
       1:04.687-1:42.884 (0:38.197)
       1:49.687-2:27.884 (0:38.197)
       2:34.687-3:15.563 (0:40.876)

channel 10:
       0:04.692-3:15.065 (3:10.373)
```

The 5 seconds period can be changed with following setting in gnmidi.ini file:

```
[Settings]
LongPauseSeconds=7
```

3.90 Show tempo changes

[in menu Analyse]

The operation collects all tempo changes within current MIDI song. Time position are displayed in format h:mm:ss.ms and tempo is displayed in beats per minute (bpm).

The information is written to a temporary text file and displayed with notepad editor.

```
c:\mysong.mid:

0:00:00.000 default tempo: 120.00
0:00:00.000 tempo: 120.00
0:00:00.000 tempo:  78.00
0:04:24.230 tempo: 120.00
0:04:41.230 tempo:  63.00
0:08:53.730 ** end song
```

3.91 Show text positions

[in menu Analyse]

The operation collects all text commands within current MIDI song and displays them with text position (h:mm:ss.ms) and type name.

Type names:

- Text
- Copyright
- Trackname
- Instrument
- Lyric
- Marker
- Cue point

The information is written to a text file and displayed with Notepad editor.

```
0:00:00.000 Trackname "Eternity"
0:00:00.000 Trackname "Soft karaoke"
0:00:00.000 Text "@KMIDI KARAOKE FILE"
0:00:00.000 Trackname "Words"
0:00:00.000 Text "@LENGL"
0:00:00.000 Text "@Teternity"
0:00:00.000 Text "@Trobbie williams"
0:00:16.153 Text "\CLOSE"
0:00:16.538 Text " YOUR"
0:00:16.730 Text " EYES"
```

3.92 Quantize pedal controllers to on/off

[in menu Modify/Controller operations]

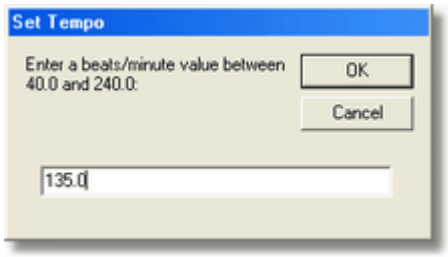
This operation modifies pedal controllers **hold** (controller #64) and **soft pedal** (#67). It maps lower controller values to **pedal OFF** and higher values to **pedal ON**.

```
  0 -  63  .................. OFF .............   0
 64 - 127  .................. ON .............. 127
```

Hint: Duplicated values are removed to reduce the number of pedal controller changes.

Hint: This operation is useful for PianoDisc and Disklavier players which may react with noises when many pedal controllers are used that are not pedal ON/OFF values.

3.93 Set MIDI tempo without changing timing



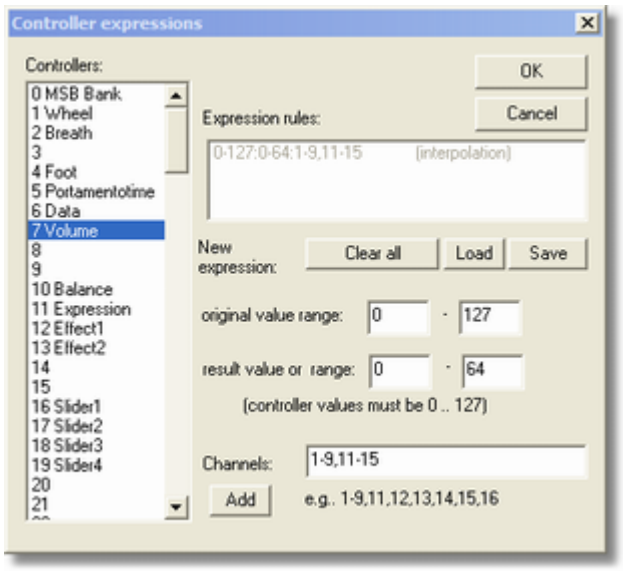
[in menu Modify/Tempo operations]

This operation replaces all tempo changes in the song by a constant tempo but keeps the original song speed by adjusting all pauses (instead of setting tempo commands).

Tempo
Enter a tempo (bpm, beats per minute) value into the edit field.

Hint: This operation is useful if your song plays in correct tempo but you would prefer that it uses an other tempo value and song still plays
with same speed.

3.94 Modify controller values by expressions



[in menu Modify/Controller operations]

The operation modifies controller values by expression rules.

Controllers
Choose a controller number that should be modified

Expression rules
displays current expression rules that are applied in this order.

Clear all
remove all expressions .

Load
load previously saved expressions from a .ctr text file.

Save
save current expressions in a .ctr text file.

Add new expression
fill original value range and result value range and then press Add to add the expression to the expression rules list.
At least one controller expression must be added that the operation can be done. The button is disabled till valid numbers are entered into the range value fields.

Original value range
enter values between 0 and 127 that the controller should have in original MIDI song that the values are modified by this expression. Values outside the given range are not modified by this expression. Value in first field must be smaller or equal to second field value.

Result value range
enter values between 0 and 127. Leave second value field empty if you want to set all original range values to same value. Here the values in first field can be higher than value in second field (for increasing the lower values and decreasing the higher values).

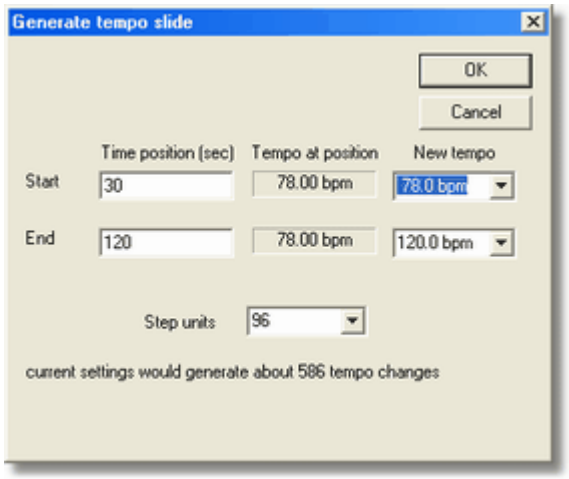
Possible expressions
- set result value to a constant value
  fill only the left field of the result value field/range, left the right field empty (e.g. set all values between 0 and 63 to value 0)
- add value to the original values
  enter values in result fields that differ to the original range values by the given value (e.g. add value 10 to all values between 0 and 63: 10 - 73)
- interpolation of value range
  enter values in both result fields, the result range width can be different to original range width (e.g. interpolate values 0-63 linear to 0-127).

Hint: The result values can also be entered reverse then the value ranges are interpolated reverse (e.g. set values 0-127 to 127-0)

Hint: the channels list and load and save buttons are available since version 2.58, before all channels were modified.

Hint: This operation can be started as batch operation. Using GNMIDI Light license batch operations are not available.

3.95 Generate tempo slide



[in menu Modify/Tempo operations]

This operation sets a series of tempo changes (increasing or decreasing) for a certain time range. This is used to increase or decrease tempo linear between two song positions.

Start time position
start time (in seconds) from beginning of the song (e.g. 10.7)

End time position
end time (in seconds) from beginning of song (e.g. 30.0)

Tempo at start position
current song tempo at start position (a hint)

Tempo at end position
current song tempo at end position (a hint)

New start tempo
tempo at start position in beats per minute (e.g. 98.0)

New end tempo
tempo at end position in beats per minute (e.g. 106.0)

Step units
distance between two tempo changes in MIDI units

Hint: Start time position must be lower than end time position.

Hint: Start tempo and end tempo must be in range 30.0-255.0 bpm.

Hint: If start tempo is smaller than end tempo then tempo will slowly increase

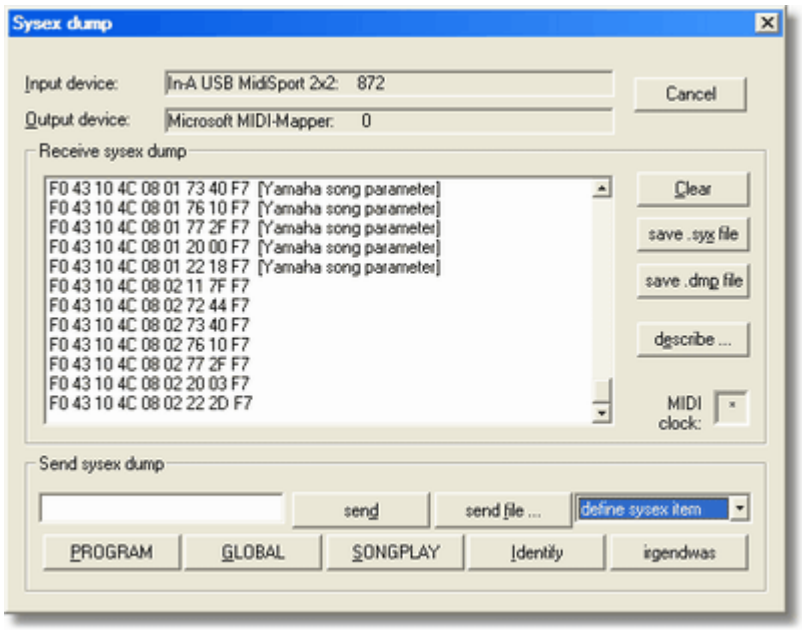
Hint: If end tempo is smaller than start tempo then tempo will slowly decrease.

Hint: If start tempo is same as end tempo then only one tempo change will be added and existing tempo changes in given range will be removed.

Number of generated tempo changes
After all required fields are entered the message below the edit fields appears which tells how many tempo changes would be generated using current settings.

Hint: Be sure that the number of tempo changes is not too high, because this could produce big MIDI file size and could be a problem for some MIDI players. If the number is too high then choose higher step units value till the number is acceptable.

3.96 Sysex Transfer



[in menu Player]

This dialog is used for **sending and receiving sysex data dumps** between computer and external MIDI devices (keyboard, sound modules ...) and reverse through MIDI cable. It has comfortable features like describing a MIDI sysex command or assign a sysex command (or sysex file) to a button to quickly send the data by a click.

Hint: key combination **Alt+X** opens this dialog.

Input device
displays the current selected MIDI input device name. It can be changed in menu Settings. The number behind the name tells the **number of bytes received**.

Output device
displays the current selected MIDI output device name. It can be changed in menu Settings. The number behind the name tells the **number of bytes sent**.

Receiving box
The list box in the Receive sysex dump group contains information about incoming sysex data (one line per sysex command).

Clear
deletes the content of the receiving box. Resets the number of transferred data back to 0.

save .syx file
saves all selected sysex data lines of the receiving box into a binary sysex data file (.syx).

save .dmp file
saves all selected sysex data lines of the receiving box into a readable and editable sysex data text file (.dmp). The .dmp file contains hexadecimal values.

describe ...
select a received sysex data line of the receiving box and describe the meaning of the sysex data. Few data bytes from beginning of the sysex command are used to recognize the meaning of a sysex in future when receiving a similar sysex again e.g. receiving song data dump ...

MIDI clock
displays a star symbol in the rectangle box when receiving MIDI data. It should blink when the MIDI connection is working.

Send sysex edit field
enter a short hexadecimal sysex command or a filename of a predefined .syx or .dmp file to send the data using send button

send
sends data entered in the send sysex edit field (left side of the button).

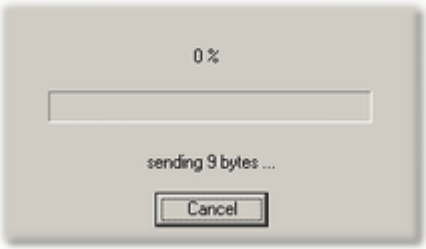
send file ...
choose a .syx or .dmp file that will be sent

choose sysex item
this combo box contains user defined sysex elements that will be sent when the item is selected. The combo box contains also possibility to define a new sysex element self.

define sysex item
user defined sysex elements can be named and assigned to a button or chosen from sysex item combo box. A dialog is opened where the new sysex element can be defined.

sysex element buttons
some space in the dialog is reserved for your own buttons that send certain sysex commands. The buttons show the name of the sysex element in the caption. Two buttons are already defined (Identify, GM Reset). They can be removed in file sendsyx.txt and replaced using define sysex item if wanted.

Sysex messages
a sysex message (system exclusive message) starts with F0 (SOX) then follow data bytes of range 0-127 and at end the sysex is terminated by F7 (EOX). Short sysex messages are used to transfer simple commands with some parameters. Large sysex messages are used to transfer songs or data dumps (which store the current settings of a device). Since the transferred data are only 7bit values, transferred data is usually encoded.



Send hexadecimal sysex message

enter a sysex command in hexadecimal representation. That is the usual form used in device manuals to list all supported MIDI sysex commands. Press send button or Enter or Return key to send the data to the output device if it is a valid sysex message F0 ... F7. It does not care if the hexadecimal characters are written in lower or upper case.
e.g. some KORG models accept following identify command

```
f0 7e 0 6 1 f7
```

If the hexadecimal sysex string is not valid then the dialog will beep and select the part of the string that looks wrong. After correct sending the connected device might respond or perform some action.

After sending the data GNMIDI will **wait certain delay time** to give the device enough time to handle the transferred data.
The delay can be modified in gnmidi.ini file by following setting:

```
[Settings]
SendSyxDelay=5000
```

Hint: If you are sure that the command is done and the device doesn't need more time then you could abort the delay by pressing **<Esc>** key.

Send sysex file (.syx)

sysex files (with file extension .syx) contain one or more sysex commands. The files can be huge (e.g. dump of all keyboard songs, all keyboard settings). Press button send file and choose a valid .syx file to send all valid sysex commands in this file to the output device. Or enter the full path of the sysex file into the sysex edit field and press send button. If the file exists and contains valid sysex data then it will be transferred.

After sending each sysex the program will **wait certain delay** to give the device enough time to handle the data.

Send readable sysex file (.dmp)

.dmp file contains hexadecimal formatted sysex data lines. The lines look like:

```
00000: F0 42 30 49 4E 06 F7
```

The hexadecimal number before the sysex data bytes (including the colon) is optional, specifying only

```
F0 42 30 49 4E 06 F7
```

in a .dmp file is allowed). GNMIDI stores the current byte position of a corresponding .syx file in this offset number when saving sysex data to a .dmp file. You can change the .dmp file without need for updating the offset values, the values are not used at reading a .dmp file.
Optionally a ! character can be used to append a comment to a line.

```
F0 42 30 49 4E 06 F7 ! GM Reset
```

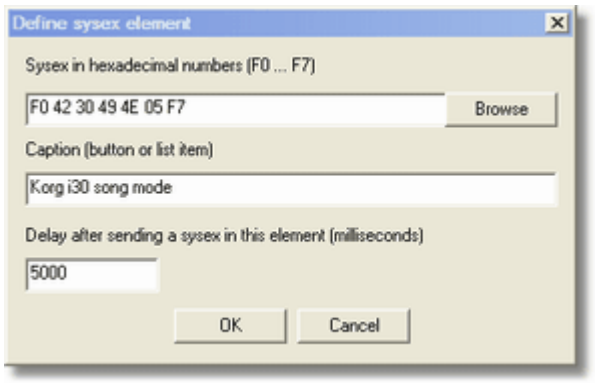
The comment is not part of the sysex data. GNMIDI inserts such comment for each known sysex. E.g.

```
00000: F0 42 30 49 4E 06 F7  ! KORG i30 change to song mode
```

Maximum line length is 1024 characters in a .dmp file. Insert line breaks between data values that the lines are short (which is easier for reading and editing). GNMIDI writes 16 data bytes per line.

When the .dmp file contains invalid formatted line then the sending will not be performed and an error message tells which line contains the problem.

After sending the data from .dmp file the program will wait certain delay after each sysex command to give the device enough time to handle the data.



Define own sysex data items

Often used sysex commands can be stored as sysex items. They can be sent comfortably by selecting a list box item or pressing a self defined button.
The second item in the send sysex combo drop down list is always "define sysex item". This item opens a dialog that defines a new sysex item that will be added to the combo drop down list and will be assigned to a send button if it is one of the first 5 items.

A sysex data item consists of fields

- sysex enter a sysex command F0 ... F7 in hexadecimal numbers or enter a .syx or .dmp filename, use Browse to select a file.
- caption enter a caption text for the combo list box item or for a button, use & prefix to declare a key shortcut Alt+(next character)
- delay enter a delay time in milliseconds that the program must wait after sending this sysex command (default 5000 ms).

After entering a complete and correct sysex data item the combo drop down list and up to five send item buttons will be updated and can be used. Currently this dialog can only be used to define new items. Modifying or deleting existing items can only be done by modifying the text file where the definitions are stored.

gnmidi.ini contains a setting that tells which file contains the sysex item definitions:

```
[Settings]
SendSyxFile=sendsyx.txt
```

The self defined items are added to the combo drop down list and first 5 ones are assigned to buttons in the send sysex area.

The text file contains the item descriptions in following form:

```
[SendSyx]
Caption=&Identify
Sysex=f0 7e 0 6 1 f7
Delay=500

[SendSyx]
Caption=&Church Sounds
Sysex=c:\sounds\sysex\organ.syx
Delay=23000

[SendSyx]
...
[SendSyx]
...
```

Hint: gnmidi.log will contain error messages if the sysex item definitions file contains incorrect definitions.

Send sysex data item from self defined button

First 5 valid self defined data items are assigned to 5 reserved buttons in the send sysex area. The caption is displayed on the button. If caption contains an ampersand character & then next character can be used for a key shortcut Alt+(next character) if this shortcut is not used elsewhere in the dialog.

e.g. caption `&Identify` will be displayed as `Identify` on the button and key combination `Alt+I` can be used to send this sysex item (if Alt+I key is not used elsewhere in the dialog as shortcut).

Send sysex data items from the user defined combo box items

All valid self defined sysex items are added to the combo box in the send area. The first item does not send anything, it clears the send sysex edit field. The second item opens the dialog to define a new sysex data item.

When opening the drop down combo box list and selecting a sysex data item with left mouse click then this item will be sent to the output device.

When selecting combo box list items with arrow keys then the item info will be copied into the send sysex edit box and then this data can be sent from the edit box (send button or Enter/Return key).

Receive sysex dumps

First choose the input MIDI device that will be used for receiving MIDI data.

The small box in the receive sysex dump area should blink, this is a proof that the input device is active (a working MIDI device is sending MIDI clock F8 or activesense FE command steadily!).

In order to receive sysex data you either need to start a send sysex dump operation on the MIDI device or send a sysex request command to the device first. Some operations on the MIDI device might automatically send a sysex notification (e.g. change to certain mode). Some MIDI devices send sysex data only when certain option on the device is set.

Every caught sysex command will be added to the list in the center, it will be displayed in hexadecimal numbers F0 ... F7 (longer sysex data will only be displayed partially, but the complete received sysex is still remembered) . While the sysex command is not complete it will be displayed ... without the F7 at end, when the sysex transfer completes then the ... F7 will be added.

The current list of sysex messages can be cleared (removes all sysex and resets transfer counters to 0). Individual sysex commands can be selected inside the list by clicking with the mouse on the line. Selected sysex commands can be removed by using clear after selection. Selected sysex commands can be saved to file (.syx or .dmp).

Describe sysex commands

Select one of the received sysex commands and click on the describe ... button. Look into your MIDI device manual to find out what this sysex command means. First 10 hexadecimal values of the received sysex are displayed in the edit field. Only a part of the sysex beginning is necessary to identify the type of command. You can delete unnecessary numbers at end of the sysex or insert more numbers if less or more bytes are significant to identify the meaning.

Special characters can be used as **wildcards** to skip meaningless numbers or numbers that are counters:

? this character inside the hexadecimal sysex part will match any byte number [0-7f]

this character inside the hexadecimal sysex part will match any byte number [0-7f] and remembers the byte number of a matching sysex at this position. The number can be added to the sysex description using the # character as placeholder

A sysex description text to a matching sysex will be

- displayed for received sysex as [...]
- displayed when sending such a sysex
- added as comment when saving a .dmp file

Currently the describe sysex dialog can only be used to define new descriptions. Modifying or deleting existing descriptions can only be done by modifying the **sysex description file** where the descriptions are stored:

Following setting in gnmidi.ini tells which file contains the sysex descriptions:

```
[Settings]
SysexDescriptionFile=describesyx.txt
```

A stored sysex description in the sysex description file looks like

```
[KORG i30 change to song mode]
F0 42 30 49 4E 06 F7
```

This will display description when this sysex is received as:

```
F0 42 30 49 4E 06 F7 [KORG i30 change to song mode]
```

For longer sysex data it is sufficient to specify beginning of sysex, the character # can be used as place holder of a number information that can be used in the description of sysex, e.g.

```
[KORG i30 style block #]
F0 42 30 49 65 #
```

will describe sysex `F0 42 30 49 65` **`02`** by following line:

```
F0 42 30 49 65 02 00 20 00 00 00 [KORG i30 style block 2]
```

Hint: Sysex data in hexadecimal numbers and their meaning can be found in manual of your MIDI device. Read also in manual to your MIDI device how to receive and send sysex dumps.

Important notice for sending sysex from file

the program waits a certain delay to give the MIDI device enough time to consummate and handle sysex data (default 5 seconds additionally to the transfer time). That is usually enough time to decode and store larger data block but can be too small if sending a series of dump requests where each might deliver much data. If the device has not enough time then it could fail to handle (larger) sysex data properly.
The default waiting time between sending sysex commands can be changed with following gnmidi.ini setting:

```
[Settings]
SendSyxDelay=15000
```

(this would set the delay to 15 seconds)

Hint: For self defined sysex items you can define the wait time individually for each sysex item.

3.97 Karaoke Editor

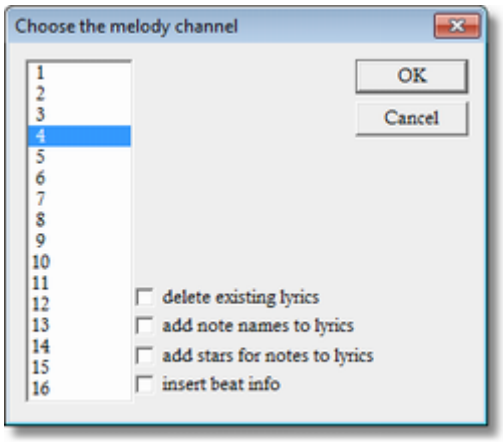
[in menu Modify]

This karaoke editor adds your song text to a MIDI file and is used to synchronize song text syllables to melody notes.
Using GNMIDI Light license karaoke editor is not available.

Hint: Unregistered demo version (to try this product) allows to synchronize first half of the song only (for testing purpose). Registered program version allows to synchronize whole song.

Important: The **melody notes** must be already available in the MIDI file, without existing melody notes this editor is not useful to synchronize comfortable, use Synchronization editor instead to synchronize song text lines.

Start the karaoke editor for the current MIDI document with shortcut key **Ctrl-K**.



Choose the melody channel
You are asked to enter the melody channel (1-16).

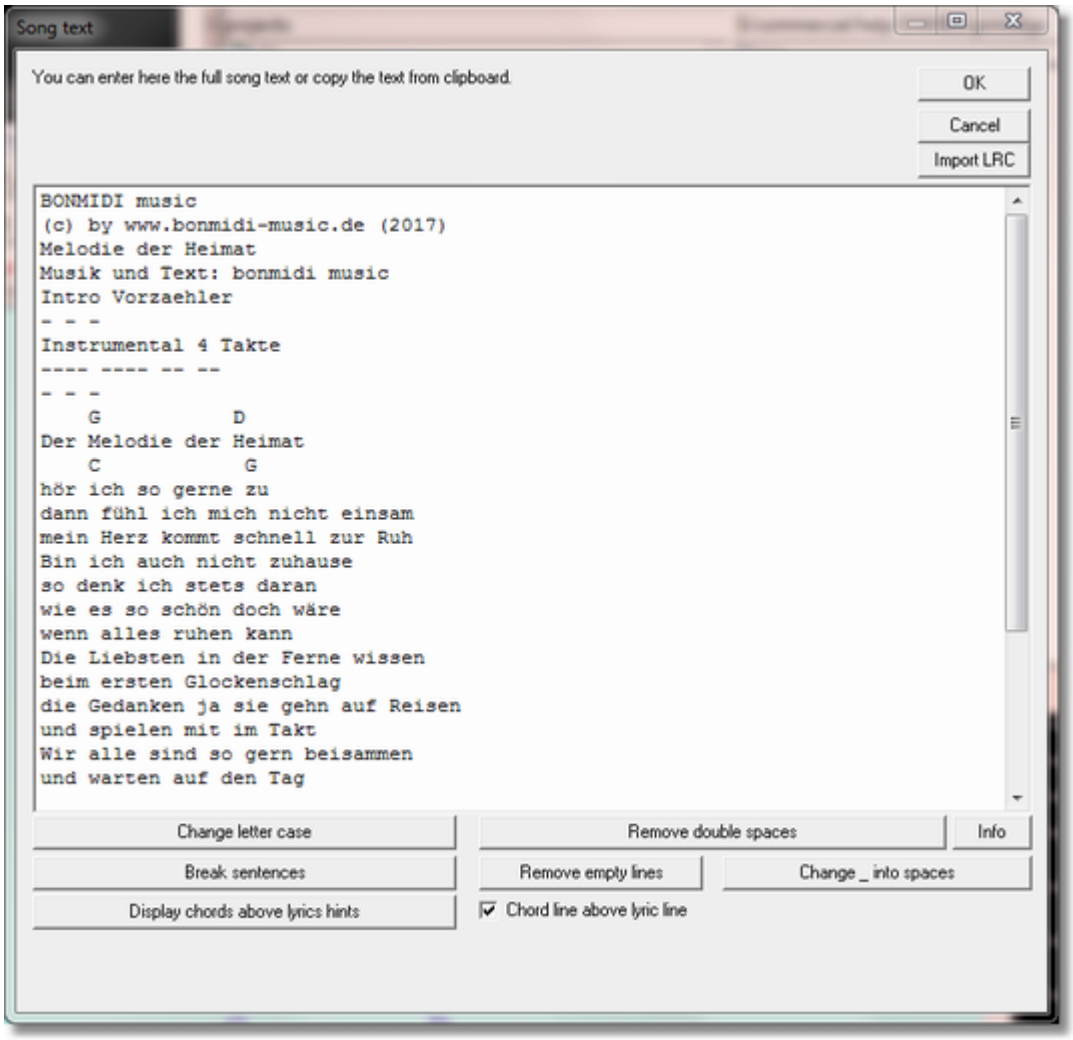
delete existing lyrics
check this option if existing lyrics should be ignored (that you can start editing from beginning with empty lyrics)

add note names to lyrics
check this option to add the note names of melody notes as default lyrics (instead of entering a song text)

add stars for notes to lyrics
check this option to add stars character * at each melody note position as default lyrics (instead of entering a song text)

insert beat info
optionally add lyrics for counting the beat with the song. This is useful when your song has no song text and needs some indication of beat.
It adds synchronized lyrics like following to the song:

```
127bpm
[001] 4/4 1 2 3 4   [002] 4/4 1 2 3 4    [003] 4/4 1 2 3 4    [004] 4/4 1 2 3 4
[005] 4/4 1 2 3 4   [006] 4/4 1 2 3 4    [007] 4/4 1 2 3 4    [008] 4/4 1 2 3 4
```



Enter song text

If the song does not contain lyrics yet a dialog is presented where you can enter the full song text. You can copy the song text from clipboard into
the edit field (open context menu with right mouse click). It is not necessary to enter the song text here, the song lyrics (syllables) can be entered later direct to the corresponding melody notes.
The entered song text will be split into words and assigned to the melody notes word by word. The words can be split into syllables later in the editor simply whenever it is required for assigning word part to a note (the grammar rules for splitting words into syllables don't apply here for this purpose).
If you enter text with hyphenation here (e.g. me-lo-dy) then the syllables are assigned to notes (without the - character) instead of the word.

Hint: you should split words into syllables by inserting - characters in this edit box whereever you probably know that the syllables will be sung by different melody notes.
This will speed up the assignment to melody notes very much since then many syllables might be already assigned to correct note.
`e.g. for e-ter-ni-ty`

Split song text into phonetic syllables
Hyphenation

For languages **English and German** the hyphenation assistance can be used. For some languages (e.g. Italian, French) only a small set words only these examples are hyphenated.

When a larger text is copied into the empty box then the dialog tries to identify the language: English, German or Other.
For text with too few different significant words the language will not be identified. The language can be changed using the drop down list.

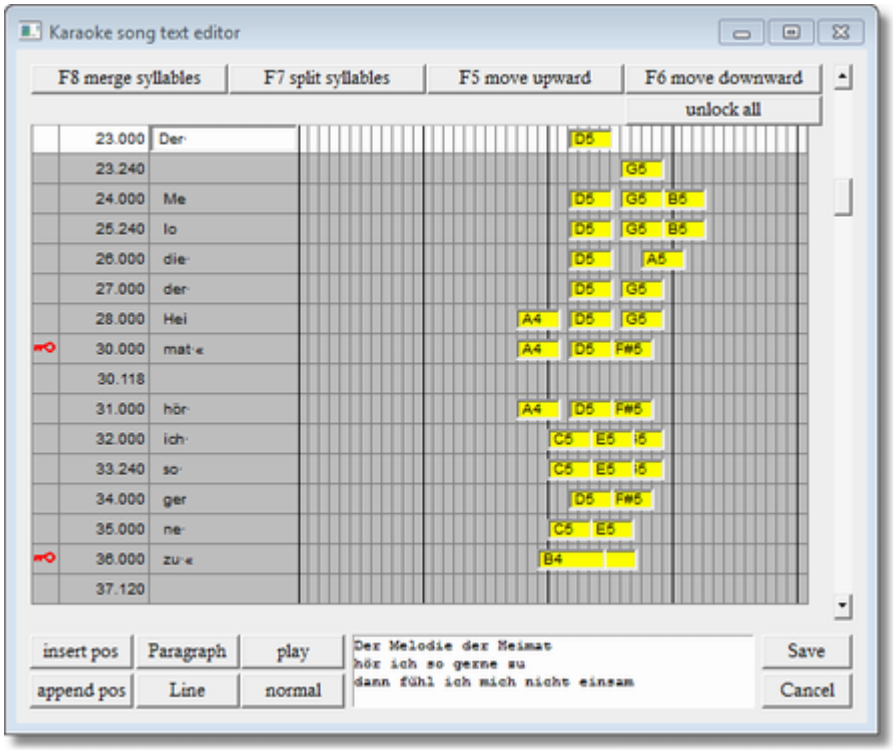
Using hyphenate syllables the text will be analysed and the most probable positions according to pronouncation will be separated by - character.
Orthography is not considered here! For English we prefer American English pronounciation.
If a text is selected only the selected part will be hyphenated.

Hint: The hyphenation files do not contain word lists but only statistically information. Not all words are considered. Some words might are differently pronounced depending on different meanings.
This assistant can only suggest a result for a word and can not identify any meaning of a word in the context.

Hint: It will not hyphenate every word correctly since it does not know the correct pronouncation for every existing word. You may send a song text to info@gnmidi.com if you think this text gives bad results during hyphenation.

You may correct wrong hyphenated words manually, mark the word and add it to the **user dictionary** of the selected language , so that in future hyphenating this word suggests your variant.

For other languages only the user dictionary is available.



The **Karaoke Editor** dialog shows a scrollable table of melody notes and available lyrics:

Columns

- LOCK/UNLOCK (key symbol)
  this small column area is used to mark a table row locked (locked when key symbol is displayed).

A locked row is protected against moving or modifying. Use this feature when a line is completely synchronized so that it won't be moved away by accident. It is not necessary to lock all lines, add the lock to the last syllable of the line. When adding a line or paragraph row then this row will get the lock automatically.

- POSITION (beat number.MIDI unit)
  this column contains a logical MIDI position for each row, positions cannot be moved, new positions can be entered between existing positions.
- LYRIC
  this column contains words or syllables that are currently assigned to a melody note or other position. During editing a row (that is not locked) can be selected and then the lyric column will display an edit field where the lyric text can be entered or modified.
  « is shown in a table cell to break the song text line at this position.
  ¶ is shown in a table cell to break a paragraph at this position.
- NOTE
  This column displays melody notes as coloured bars which display the note name. The horizontal beginning position of the bar gives a visible hint about the note height (it starts more right if note is higher). The bar width gives a hint about the note duration (the wider the more longer the note is played).

Hint: Click (with left mouse button) into LOCK column will toggle the locking state. Click into the other columns will select the row and offer an edit field for changing the lyric.

Scroll bar
Use the scroll bar to show the rows before or after current table page. Use the arrow keys UP, DOWN, PAGEUP, PAGEDOWN scroll the rows manually.

Text preview
The text currently contained in the page rows is displayed in preview area below, this displays also the line and paragraph breaks, that helps to find out where line or paragraph breaks are still missing.

insert pos
inserts a new row before current selected row. It suggests a position in the middle of previous and current row position. The position must be specified in beat.unit e.g. 16.32 . Use this operation when there are syllables in the song text that should be assigned to a position between two melody notes.

append pos
inserts a new row between current selected row and next row.

Paragraph
appends a paragraph end marker ¶ to current row text. A paragraph separates melody lines that don't belong to same verse. The row will be locked to prevent it from being moved without intention. Add paragraph when a song text line is correctly synchronized.

Line
appends a line end marker « `to end of` current cell text. A line separates two melody sentences that belong to same verse. The row will be locked to prevent it from being moved without intention. Add line when a song text line is correctly synchronized.

play/stop
starts playing from song position of the top row of this page till stopped by user or end of song. The rows will be highlighted at matching row positions. The playing speed is controlled by option normal/slow. The button caption toggles between play and stop depending on the current playing state. *Warning:* Space key can't be used here to start/stop the player since space key is used to enter the space character into the edit field.
Hint: Playing automatically stops when using an edit operation or selecting a row (by click on the row).

Hint: Player starts to play directly at the time when the melody note first on the current page begins. You can force the editor to play some time (in milliseconds) earlier before top row note begins with following gnmidi.ini setting:

```
[Settings]
KaraokeEditPlayerStartDelay=2000
```

normal/slow

the option defines the playing speed when using the play button. Toggle between normal speed and slow speed. Normal speed will play the MIDI song in original tempo. Slow speed is defined to default 60% of original song tempo.
This setting can be changed in gnmidi.ini file:

```
[Settings]
KaraokeEditSlowPercent=60
```

Save

This button saves current karaoke song to a temporary MIDI document (opens a document window). Don't forget to use Save or Save As operation in the file menu. The resulting MIDI file will be in format 0 and contain standard MIDI lyrics. Use karaoke conversion or .kar conversion operations to change the lyrics format.

Cancel

use this button only if you want to abort the editing without saving the modifications. You are asked to confirm this decision. This is useful when you are trying the Karaoke editor functions.

Move upward

moves the text in current selected row and rows above up one row till an empty row is found. The operation will fail if no previous empty row is currently available till first row. Key F5 can be used for this function.

Move downward

moves the text in current selected row and rows below down one row till an empty row is found. The operation will fail if no next empty row is currently available till last row. Key F6 can be used for this function.

Hint: GNMIDI appends empty rows at end of song so that there will be new space for moving downward when necessary. If there is no free row till next move downward, GNMIDI will try to generate a new free row by merging the last to syllables before next locked row.

Split syllables

splits a word in current row at cursor position into two syllables. The syllable on the right side will be moved down one row (including text in next rows) till a free row is found. The operation will fail if no next empty row is currently available till last row. Key F7 can be used for this function.

Hint: Each time when you split a word into two syllables GNMIDI searches for same words after current position and tries to split it into syllables same way (only possible if not locked and free row available till next locked row). This auto split syllables behavior can be turned off with following gnmidi.ini setting:

```
[Settings]
KaraokeEditAutoSplit=0
```

Merge syllables

merges the text of current row and next row to one text in current row. The text of next row will then be empty. The selection moves down to the empty row. Key F8 can be used for this function. This function is also necessary when moving text fails, because the function generates empty rows after merging text rows.

unlock all

removes red key lock symbols on all rows
Warning: when using move down or move up this will move even lyrics that you might have already correctly assigned to note row.

Editor Modes
The Karaoke editor has two modes that can be switched any time by pressing play/stop button:

- **PLAY MODE**
  this mode is used to check if the synchronization is correct by playing from beginning from a song position where top row is located. Define the playing speed with normal/slow button and play/stop the current song part. The **rows are highlighted** white whenever the song reaches a row position and rows are automatically scrolled into view.
- **EDIT MODE**
  this mode is used to **enter or modify lyrics** in rows, move rows up or down insert/append new rows, append line or paragraph breaks, split or merge syllables. Select a row to perform a function at this text or position.

Enter/modify song text
click with left mouse button into a row (columns 2-4) to select the row for editing. An edit box will be displayed in this row inside column 2 that contains the current lyric text. You can enter new text or modify existing text in this edit field. Use arrow keys UP/DOWN to get directly into previous/next row edit box.

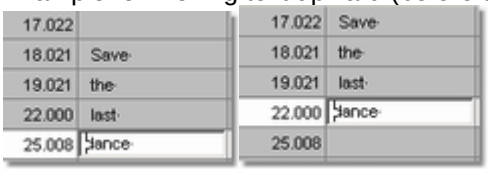
Hint: While entering text it is not important that words/syllables are already entered into the correct synchronized row. It is easier to synchronize the entered text later where you can split/merge syllables, move lyrics up or down later quicklier.

Important: **Space** characters are shown with a **point in the middle of the character**, so that the are visible during editing. Spaces between words should be always at end of a syllable. Look into preview text area so that you can see if words are correctly separated by spaces and that no space is in middle of one word that is split into syllables.

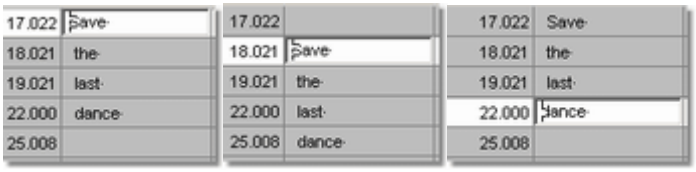
Hint: Don't enter special formatting (e.g. .kar format) instructions except line break « and ¶ at end of syllable. The save operation will save in a standard lyrics format that can be converted with GNMIDI later (to .kar or other karaoke formats).

Synchronizing song text
for synchronizing syllables you need to move the lyric syllable to the row of the corresponding melody note or a self defined position between two melody notes. Notes can't be moved only the text content can be moved up or down into a row that contains the melody note. For moving a row with empty syllable/word must be available in this direction. Locked rows can't be moved, an error message will be displayed when no available free row is found that is necessary for moving.

Example for moving text upward (before and after pressing key F5):



Example for moving text downward (before and after pressing key F6):



When **no empty row** can be found or generated (or a locked row prevents from being moved) then following message will be displayed:

```
No free element found that elements can be moved.
```

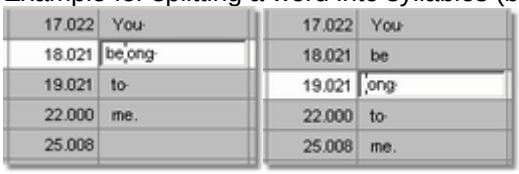
Hint: **Lock some rows** that contain already synchronized text so that this text is not again moved away by accident by a further move operation.

Hint: It is not necessary that all melody notes have corresponding syllables, sometimes a syllable is specially interpreted by a series of notes. Rows that don't contain text will be ignored at save. Look also at position information, when two positions are very close then it can be better that only first of these rows contains text and second is left empty.

Split words into syllables

First select the row that contains the word, move the cursor in the edit field to the position between the two characters that are to split into two syllables. Press F7 to split the current word into two parts. The right half of the word will be moved down into row below. For successful splitting it is necessary that a row with empty content is available somewhere down, so that next row content can be moved down.
If no empty row is available then this will be displayed in an error message. Then you should look at rows below and merge some rows so that at least one free row will be generated. It is possible that the MIDI song contains less melody notes than are necessary for synchronizing all verses. In this case it is necessary to delete some verses.

Example for splitting a word into syllables (before and after pressing key F7):



Hint: Since splitting moves down rows content it is recommended to lock important rows behind current row that are already synchronized.

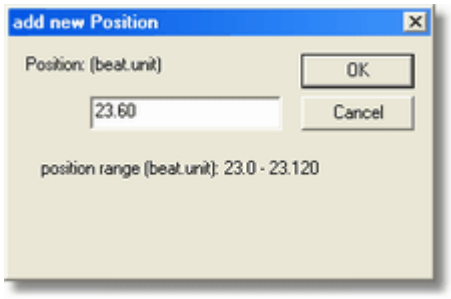
Hint: Since the splitting is only done for synchronizing spoken parts of word to notes you don't need to follow any rules of language grammar here. Split wherever you think that word parts are played by different notes.

Hint: After each splitting of a word GNMIDI tries to find identical words behind current position and splits it same way if possible.

Merge syllables to a word or sentence

select the first row of two adjacent rows that contain text that should be merged into one row content. Press F8 to merge the text of current and next row.

Example for merging two rows to a word or sentence (before and after pressing key F8):



Insert a syllable between two melody notes

Usually song text is directly assigned to melody notes, but in certain cases there is no melody note available for a syllable (e.g. the song author made a little mistake and forgot to play a note, or the author uses the pitch bend wheel to play certain parts).
Use the insert or append pos buttons to define a new position between the two melody note positions. A new row will be inserted where you can enter the text for this position.

Hint: Currently no modify or delete position function is available. Leave the row empty if the position is not useful for the syllable. Empty rows are not saved.

Add line breaks or paragraph breaks behind synchronized text lines

Each song text lines should be ended with a line break or a paragraph break. Look into the preview text area, if the lines are too large then you should break them directly after last syllable of the sentence. Use line break for lines inside a verse, use paragraph break after last line of a verse. Rows that contain a line break or a paragraph break will automatically be locked with the key symbol. That prevents that the synchronized line might be moved by accident. If the line break was added wrong then unlock this row (click into first column of this row) and delete the « character manually.

Hint: Don't use paragraph break and line break « at same position.

Hint: Unlock all locked lines at once on your own risk with key F9. Use that only when you want synchronize whole song again from beginning, e.g. if you started with line synchronization editor and now want to synchronize syllables. It is dangerous to unlock synchronized lines, that lines could be moved unwanted to wrong positions.

Save a modified karaoke song

First exit the dialog with save button. That will create a MIDI document window in your GNMIDI environment. Use Save As operation to save the MIDI file.

Hint: The dialog can be resized with help of dialog frame and title bar.

Change the fonts:

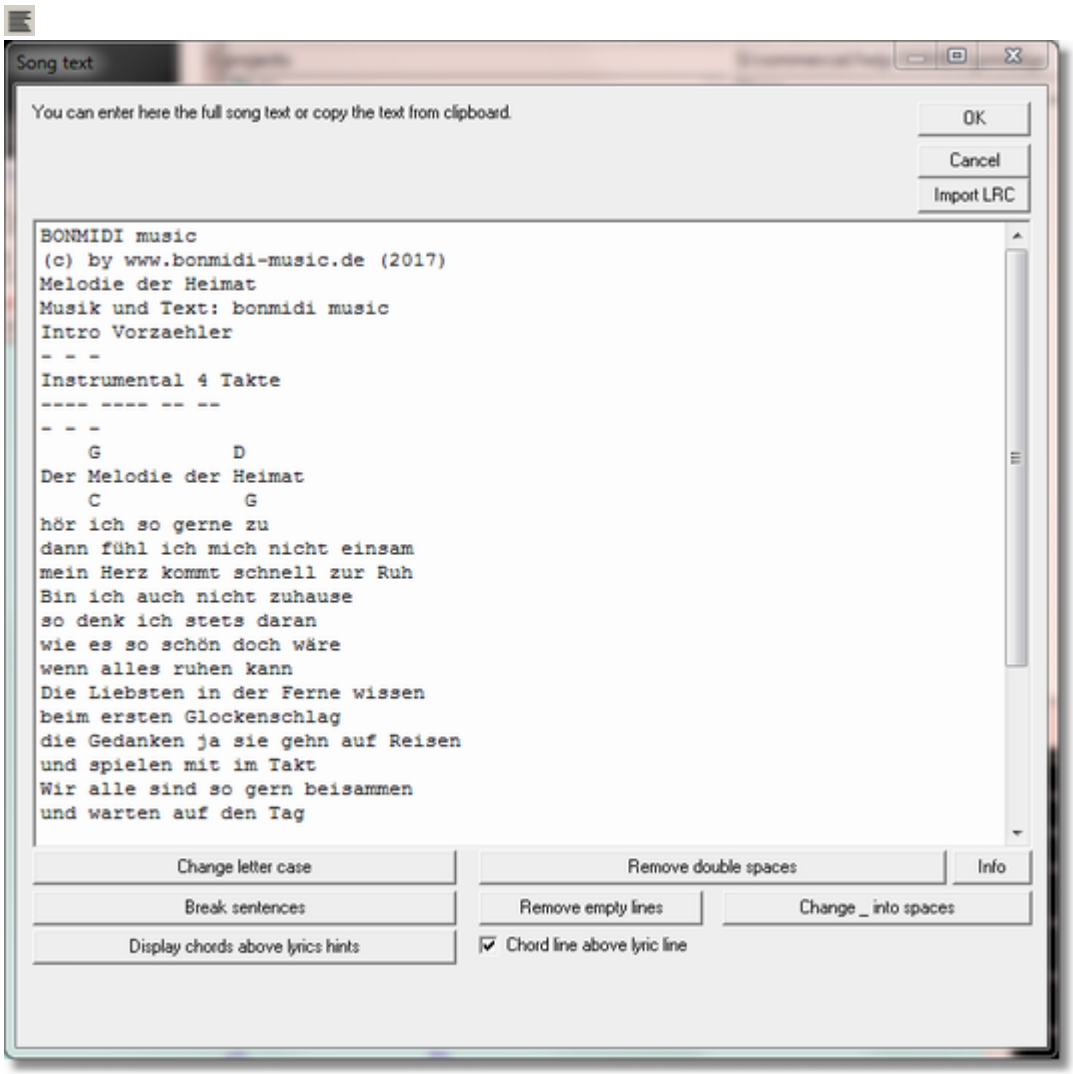
The font names and font sizes used by the karaoke editor can be changed by a setting in file GNMIDI.ini (a text file in your documents folder)

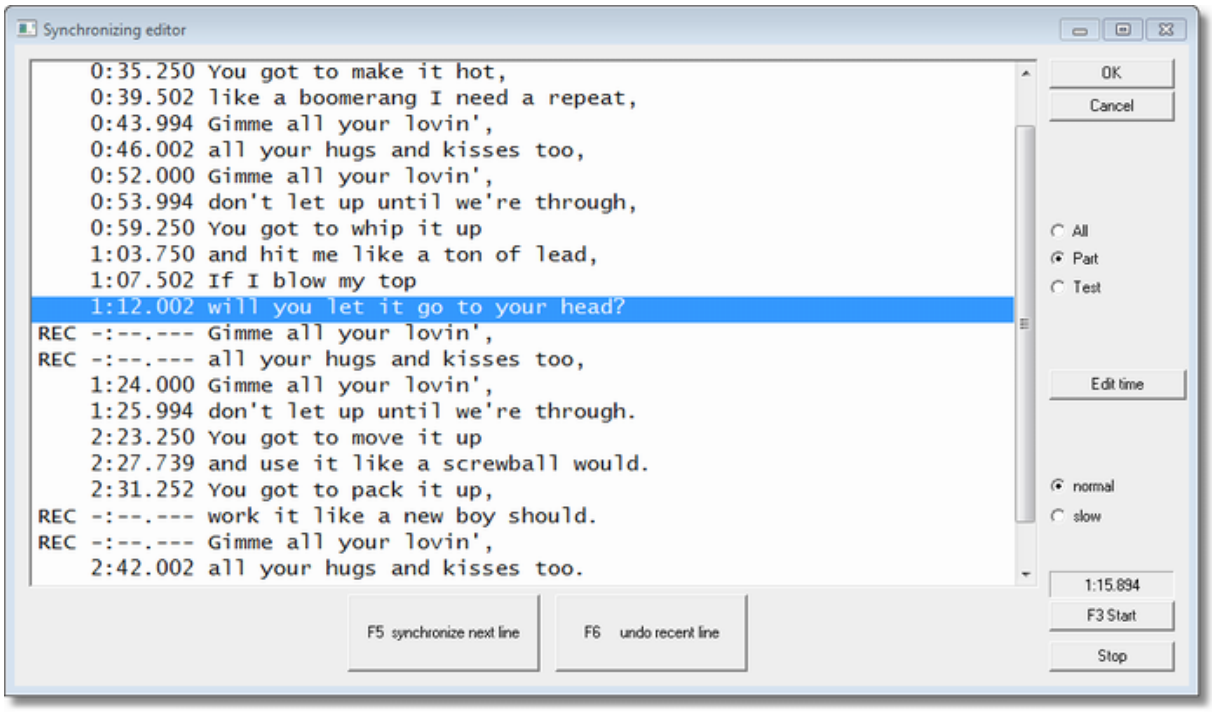
```
[Settings]
KaraokeEditorFontname=Arial
KaraokeEditorFontsize=11
KaraokePreviewFontname=Courier New
```

```
KaraokePreviewFontsize=10
```

The preview font is used in the box at bottom side. The editor font is used in the list (syllables and note names).
The [Settings] line already exists in GNMIDI.ini, insert the lines that you want to change if not existing.

3.98 Synchronization Editor





[in menu Modify]

The Synchronization Editor is used to enter song text line and synchronize the song text lines to MIDI or MP3 song positions. Synchronization can be done comfortably during playing the song.

Hint: If you plan to synchronize syllables, you should directly use Karaoke Editor. For beginners and for applications where text line synchronization is enough the synchronization editor is simpler and quicker to use.

Demo limitation

In unregistered demo version (which is for testing only) you can only synchronize first half of the song. Registered program version synchronizes all song text lines.

Choose a MIDI document window and start the synchronization editor. Shortcut key Ctrl+Z can be used.

Warning about synchronized syllables that are probably synchronized

If the MIDI file contains song text that is already split into syllables and probably synchronized syllable by syllable then a warning appears. You should abort if you want to keep the syllables and better use karaoke editor to check and modify syllables synchronization. If you continue then the song text will offered for synchronization line by line. Unmodified lines will keep synchronization time of first syllable in this line, you don't need to synchronize them all again if they are already ok.

Enter the song text

First a dialog is displayed where you **must enter the song text**. Existing lyrics will be collected from the MIDI file and displayed in the edit box. You can also copy text from clipboard (use right mouse button and choose paste). Next dialog does not contain possibility to enter or modify lyrics, so that must be done in this dialog.
Lines may optionally begin with a time stamp (format is [minute:second.secondpart] or [minute:second] or [second.secondpart]) if the synchronization time is already known.
e.g.
[0:27.32] means millisecond 27320

Import LRC loads a lyrics file with optional time stamps at beginning of lines. Syllable time stamps (format <...>) are ignored.

Hint: use 3 digits if you want to use milliseconds e.g. [3.045] is millisecond 3045
Hint: unsorted time stamps are ignored
Hint: Import LRC can be used to copy synchronized lyric lines from an other song that uses identical timing (e.g. MIDI file rendered to mp3)

Change letter case Words that look strange formatted are changed to lower case, if a sentence begin is found then first character changed to upper case.
Remove double spaces Double spaces have a special meaning in GNMIDI when displaying synchronized lines. They are used to format a chord line above next lyric line by spaces. Double spaces should not be used in text if not using this feature.
Info button explains why double spaces removing could be necessary in some cases.
Break sentences break lines in selected area after typical sentence characters . ! ? ; : -
Remove empty lines remove lines that are empty or contain only whitespaces in selected area
Change _ into spaces translate underscore characters to space characters
Display chords above lyrics hints show hints about currently detected chord lines (<OVER> or <LINE>) and display spaces as middle dots
Hide chords above lyrics hints remove hints about currently detected chord lines (<OVER> or <LINE>) and show spaces instead of middle dots
Chord line above lyric line check box to identify chord lines for display above lyric lines, set check off to turn that feature off

Hint: most above functions can be used for selected text or whole text

Hint: Enter lyrics dialog uses font Courier new by default. It should be used with a monospaced font (all characters have same width) that the feature to position a chord line above the correct lyric words in next line can be supported.
The font can be changed manually in gnmidi.ini settings file in your documents folder using Notepad editor by adding:

```
[Settings]
EnterLyricsFontsize=8
EnterLyricsEditorFontname=Courier New
```

If using a non-monospaced font then a warning will be written into log file that the feature chord line above lyric line will not work.

All
when this setting is chosen all lines will be synchronized. Use this to restart synchronization from beginning.

Part
when this setting is chosen only lines will be synchronized that are set to recordable (REC). To set lines to state recordable you need to select one or more lines with left mouse button and then click on option range. Lines without times (--:--:---) don't need to be selected, those lines are automatically treated as recordable (REC).

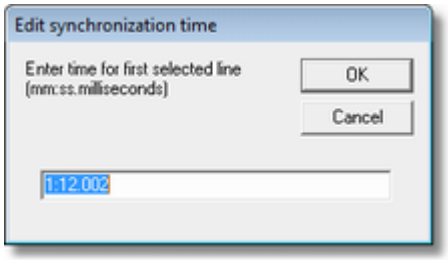
Hint: It is important that you first click into lines in the list box and then on the Part check box (even if it is already selected). This will
set the selected lines to REC. In part mode the player will start some seconds before the first REC position, so you can easily repeat
recording from other positions.

Hint: Double click with left mouse button into the list box to unselect all list items at once.

Test
when this setting is chosen all lines are set to not recordable which prevents from modifying current synchronization during playing. Use this to check synchronization of lines and find out which lines should be synchronized better.

Synchronization list box
The list box contains song text lines. In front of every synchronized line there is a time position specified in minutes:seconds.milliseconds e.g. 0:16.168 .
Not yet synchronized lines display a --:--:-- in front of the lines. It will be replaced by the time when clicking F5 during synchronization.
In front of the lines that are intended to be synchronized at next start (F3) you find word REC (recordable). All other lines don't have this state, so they won't be changed during playing. Lines without time --:--:-- are recordable (except in mode Test).
Lines that contain <space><space> are treated as chord line above next lyric line and get automatically time of the following lyric line. They will have word OVR in the list.
Hint: Exception are lines that end with <space><underline> are not treated as chord lines.
Hint: this feature could be turned off in menu settings

Start (key F3)
starts to play the MIDI song from beginning in selected speed and listens on user key input for synchronizing next line. Using Start will forget all previous synchronization times from last Start.



Edit time
sometimes it is enough to specify a time or change a time directly. First select a line and then use the Edit time button.
The dialog expects a time format minute:second:millisecond or minute:second .
All lines must have increasing sorted times so therefore the specified time must between the times of previous lines and following lines.
If there is no space for the wanted time then wrong times in other lines must be set to REC first using Part button (which deletes the times of selected lines).

Time display
current playing time will be displayed during playing of the MIDI song, it is not displayed when stopped.

Stop (key F3 while playing)
stops playing the MIDI song and aborts synchronizing. All synchronized song text lines till now are remembered and can be saved. Use again Start when you want to forget the recent synchronization try and try it again from beginning. While the song is playing you can use key F3 to stop.

normal / slow speed
select normal or slow speed for playing the MIDI song in original tempo or in a slower tempo (default 60% of the original speed). This setting can be changed in gnmidi.ini file:

```
[Settings]
KaraokeEditSlowPercent=60
```

Synchronize next line (key F5 while song is playing)

press the Synchronize button or key F5 every time when the beginning of next line is currently about to be sung. The next line will be selected and the time position will be displayed in the list box, if the next line is recordable (REC) otherwise the user command will be ignored with a warning beep.

Undo recent line synchronization (key F6 while song is playing)
when **key F5 was pressed too early** (before the next song text line really starts) then you can correct this when you are quickly. Press Undo button or key F6 to set the synchronization status of previous synchronized line back to --:--:-- (not synchronized yet). At the correct position synchronize this line again with key F5. If recent line is not recordable (REC) then this operation is ignored with a warning beep.
Hint: When you pressed key F5 too late then undo doesn't help, you can either accept the wrong position if it was not too late or stop synchronization and restart again from beginning of the song. Use option Range to modify synchronization of some selected lines instead of option All where you need to synchronize all lines again.

Save song with synchronized lyrics
Save button integrates standard MIDI lyrics at synchronized times into a MIDI file. All lines that are not synchronized will get the time of the last synchronized time. Save only creates a temporary MIDI document, you need to use Save As operation to save the modified song to a MIDI file.
In demo only half synchronization of song will be saved, register to be able to save full song synchronization.
Results will be saved in standard MIDI lyrics format, use .kar conversion or other lyrics conversion operations to convert the file.

Synchronize MP3 song text
For playing MP3 song with internal GNMIDI player a working MCI MP3 driver must be installed in Windows system.
After synchronizing lyric lines the lyrics are stored in a temporary copy of the original MP3 file. GNMIDI and newest versions of Windows Media player can play MP3 with integrated synchronized lyrics.
Hint: Since v3.14 GNMIDI supports also Lyrics3 song text format. In settings menu you can select which mp3 song text formats should be generated.

New feature:
.add chord lines or lines with instructions which are displayed directly above the lyric words.

3.98.1 chord lines simply edited with synchronisation editor

GNMIDI 3.12 and later contains a highlight feature for editing synchronized chord and lyric lines.

Normally the synchronisation editor is used to enter lyric lines and synchronize them to the music times.
Now you can enter a chord line above a lyric line or other instructions above a lyric line.
The **enter lyric dialog uses a fixed character width font** so that you can position the chords directly above a word by **inserting space characters**.

```
G              D
Der Melodie der Heimat
C               G
hör ich so gerne zu
C                       G
dann fühl ich mich nicht einsam
C                          D
mein Herz kommt schnell zur Ruh
```

The synchronisation editor and the GNMIDI karaoke display features assume a line that contains a

series of spaces to be a line that should be **displayed above the following lyric line**.
The synchronisation editor will assign the synchronisation time of the lyric line also to the chord line. Other software or hardware would display these words positioned by space characters at totally wrong positions when using a nice display font. The karaoke display algorithm of GNMIDI uses the fixed space **layouting of the chord line** above the lyric line to display the chords above the right words **(by inserting empty pixels**).

Hint: if a chord line contains only single space between the chords or no space (single chord in line) then **append two or more spaces at end** of this line so that it is assumed to be joined with next line.

Hint: if a line contains double spaces <space><space> but is not intended to be aligned to following line then add <space><underline> so that it is not treated as joined line.

```
    2. Refrain _
G               D
Der Melodie der Heimat
```

Hint: several chord lines before same lyric line are not supported

Hint: This method is not used when song text is synchronised syllable by syllable or chords are synchronised with their own synchronisation times.

Hint: this feature could be turned off in menu settings

Hint: In the enter lyric dialog of synchronization editor there is a button that displays which lines are **<OVER>** and which lines are <LINE>.

Hint: If two <OVER> lines are before next <LINE> then the first <OVER> will not be formatted like the <OVER> line directly before the <LINE> (only one <OVER> is currently supported).

3.99 Seek long notes in song

The operation searches notes within song that play a note for at least 5 seconds or are incomplete (NOTE ON or NOTE OFF command missing). The result is written to a text file and displayed by notepad editor. This operation useful when think that there are playing notes too long and want to find out the reason.

```
c:\mysong.mid:
channel  6 note  c5 vel  91 started at unit 157200 plays for duration 11.054 seconds
channel  6 note  d5 vel  91 started at unit 159120 plays for duration  5.048 seconds
channel  6 note  f5 vel  89 started at unit 159120 plays for duration  5.580 seconds
channel  6 note a#5 vel 101 started at unit 159120 plays for duration 41.038 seconds
channel  6 note d#5 vel 102 started at unit 162480 plays for duration  9.554 seconds
```

Note start position is displayed in MIDI units. With MIDI calculator you can convert MIDI units to time or beats or measures.

The 5 seconds period can be changed with following setting in gnmidi.ini file **in your personal *My Documents* folder**:

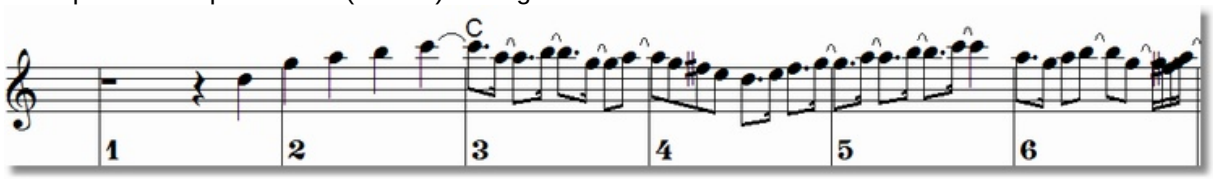
```
[Settings]
LongNoteSeconds=10
```

3.100 Fit improvisation to a score sheet

[in menu Modify]

An **improvisation** is a music piece that is played live without metronome.

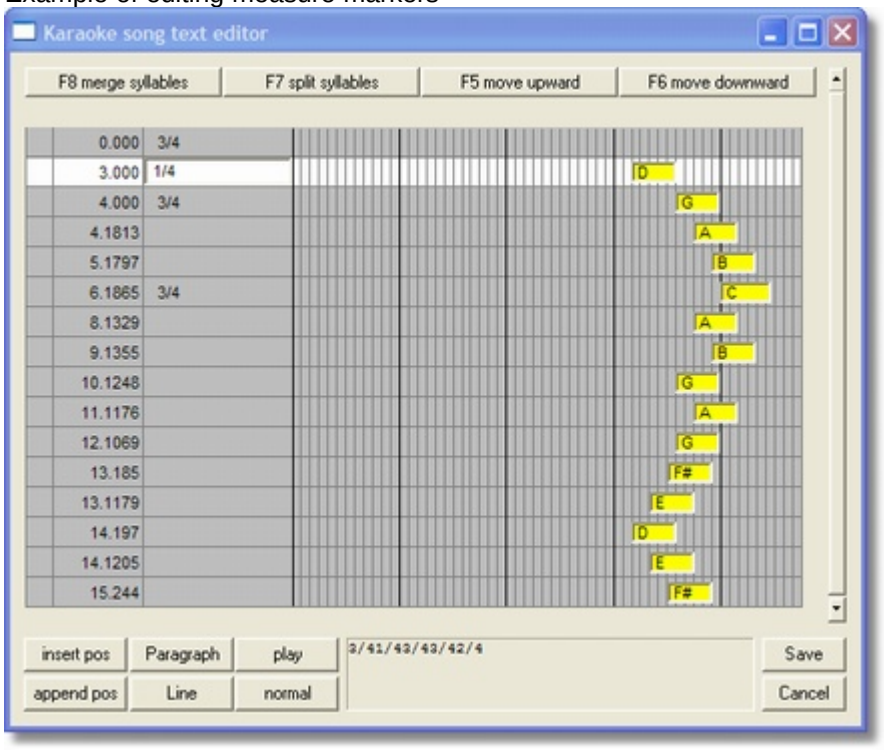
Example of an improvisation (Mozart): in original scores it was 3/4



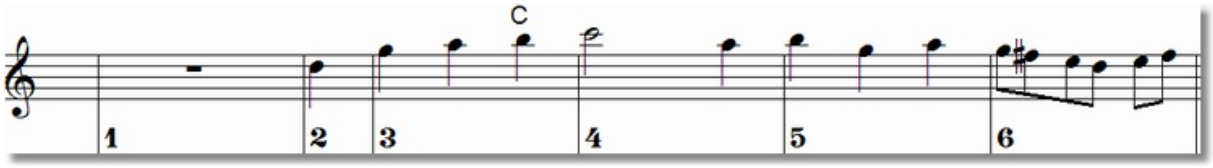
The MIDI song does not contain enough information to display the song correct as a score sheet (missing or wrong measures). Minimal tempo differences or playing errors cause unwanted score display, the notes are placed at not intended positions and that makes the score sheet hard to read.

This operation does not try to correct errors, instead it uses given **measure markers** to move these tact markers to correct logical score sheet positions, without changing the playing noticeably. The score display will become better the more measure markers are placed precisely.

Example of editing measure markers



Example of an improvisation that was fit to score sheet with this operation (while the MIDI song still plays identically after the strong modifications!)



Format of measure markers

Measure markers must be entered into the MIDI file before using this operation. Measure markers are simple text command, which must be inserted at correct position where measures start and which contain only measure meter information:

Example
4/4
3/4
1/4
6/8

Hint: The measure information must not contain any characters or other symbols in front of this number format (e.g. T4/4 or 4:4 is not valid). The measure information may contain other text (separated by a space) or line separator behind the text (e.g. 1/4 Measure 1). Only one measure information is allowed per field and position (4/4 3/4 is not valid).

Insert measure markers

Inserting measure markers works best with karaoke editor, when the song does not contain lyrics yet and when you choose a channel that contains notes which can be used easy to identify a position where a measure should start. Simply enter the measure information (e.g. 4/4) at the position where the measure (and often also a note starts). If there is already a karaoke text at same position then it is best to insert a new karaoke text position with same position where you can enter the measure information.

Also synchronization editor can be used to enter measure markers, but make sure that you click at time where the measure starts very precise, which often does not work so exact. If the song already contains karaoke text then this text might disturb synchronization of measures.

Hint: You need to enter at least two measure markers in a row, that GNMIDI can estimate the measure duration and guess missing measure markers self. Optimal results require inserting all measure markers. When tempo of song changes then additional measure markers are required that GNMIDI knows the new measure duration.

Hint: If no or too few measure markers are inserted then a message will be displayed that more measure information is necessary.

Hint: GNMIDI tries to modify logical improvised measure length to correct measure length by changing tempo of this measure without changing duration of the improvisation measure MIDI Tempo is limited to a certain value range 8-500 bpm. Wrong or extreme measure information could lead to extreme cases where a very short part must be extended to a very large part (e.g. played part 1/96 to displayed part 4/4) which could only be reached using a tempo range outside of the valid tempo range, and so in this case GNMIDI must give up. Try to play the improvisation very close to the original or imaginary score sheet and try to place all measure markers exactly. Don't force to fit very short played parts into large displayed measures and reverse.

Hint: The first measure and also an incomplete rest of a measure between start of midi song and first measure info is very critical. Also this remaining part must fit into a valid measure, a score sheet does not allow unaligned beginnings. For this area it is recommended to use shorter measure length (e.g.

1/4) to reduce the risk of tempo range overflow.

Warning: MIDI file assumes measure info 4/4 for any part before first explicit set meter info!

Testing

After generating a new MIDI song load this new file into your MIDI sequencer or any software that can display or print score sheets from midi file input.
Check if the notes are displayed at correct positions, if you have the original score sheets then compare it with the displayed results.
If notes are positioned inexactly within a measure (e.g. notes that should be at beginning of a measure are starting early or late) then try to correct the measure markers for this measure position (move or modify existing measure information or insert a new correct measure marker if it was missing at this position).

Hint: GNMIDI does not display score sheet, you need a 3rd party MIDI software that is able to display and print score sheets.

Hint: Real playing errors are not corrected with this operation. If two notes should be displayed at same measure position but were played with inexact timing then this error could be visible in score sheet after fitting. Try to play more precise or correct the playing errors with a MIDI sequencer or using ASCII conversion.

3.101 Monophon channels

[in menu Modify/Note operations]

Polyphon channels play more notes at a time. This operation reduces and **removes overlapping notes** so that each channel plays only single note at a time. Pedal MIDI controllers are removed, since they influence polyphony too.

Mono phon melody is used in some applications like phone melodies, search by melody (e.g. parsons code).

Hint:
Create a MIDI file from a polyphon MIDI song by removing all channels except melody channel and then use this operation to ensure that the melody channel will be monophon.

3.102 Quantise notes

[in menu Modify/Note operations]

Quantise
This operation aligns command positions and note duration at given raster. Quantising is usually used to correct minimal imprecise note starts (which are caused during live playing).

Channels
This operation can be used with all channels or single channel or some channels at once. Select the channel numbers in the list box.

Positions
The command positions will be set to closest matching raster position. Raster precision will be specified in score sheet pause units (e.g. 1/64). This is optional and set by default.

Durations
Note durations will be aligned to an own raster that is given in score sheet note units (e.g. 1/16). Chosen raster size is minimal note duration too. This is optional and off by default.

Hint:
Pitchbend commands are not aligned. Use only small raster size when a channel uses Pitchbend commands.

3.103 Compare MIDI files

[in menu Analyse]

Choose second MIDI file

Open dialog is used to choose a second MIDI file.
Active MIDI file (in current active window) will be compared against the second MIDI file.

Result
A text file will be generated and opened as result that contains two columns (one per file) with different MIDI commands.

Hint: Comparing could take longer if MIDI files are not very similar (since all differences will be listed in

the file).

Example: only tempo was changed in a MIDI file

```
FILE c:\midi\deutsch\LIEBESSP.MID       FILE C:\Temp\new00170.mid
Trk  1       Unit     0 Tempo 100.000 bpm
                                        Trk  1       Unit     0 Tempo 110.000 bpm
```

Example: Some text was modified in a file

```
FILE c:\midi\deutsch\LIEBESSP.MID       FILE C:\Temp\new00176.mid
Trk  3       Unit     0 Text type 1 length 56 @Tsequenced by Günter Nagler (...
                                        Trk  3       Unit     0 Text type 1 length 46
@Tsequenced by Günter Nagler
Trk  4       Unit     0 Text type 3 length 6 Melody
                                        Trk  4       Unit     0 Text type 3 length 7 Melodie
Trk  5       Unit     0 Text type 3 length 9 whistling
                                        Trk  5       Unit     0 Text type 3 length 8 Pfeiffen
Trk  6       Unit     0 Text type 3 length 13 horse running
                                        Trk  6       Unit     0 Text type 3 length 12
Pferdegalopp
```

Example: PSR META chords were added to a file

```
FILE c:\midi\deutsch\LIEBESSP.MID       FILE C:\Temp\new00178.mid
                                        Trk 14       Unit     0 Text type 3 length 48 Chords
guessed by {GNMIDI}
                                        Trk 14       Unit     0 Meta  type 7F length 9  43 7B
00 58 46 30 31 00 11
                                        Trk 14       Unit     0 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit   480 Meta  type 7F length 7  43 7B
01 45 00 7F 7F
                                        Trk 14       Unit   576 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit   672 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit   768 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit  1056 Meta  type 7F length 7  43 7B
01 45 00 7F 7F
                                        Trk 14       Unit  1152 Meta  type 7F length 7  43 7B
01 37 00 7F 7F
                                        Trk 14       Unit  1248 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  1344 Meta  type 7F length 7  43 7B
01 37 00 7F 7F
                                        Trk 14       Unit  1440 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  1536 Meta  type 7F length 7  43 7B
01 37 00 7F 7F
                                        Trk 14       Unit  1632 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  1728 Meta  type 7F length 7  43 7B
01 37 00 7F 7F
                                        Trk 14       Unit  1824 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  1920 Meta  type 7F length 7  43 7B
01 37 00 7F 7F
                                        Trk 14       Unit  2016 Meta  type 7F length 7  43 7B
01 00 22 7F 7F
                                        Trk 14       Unit  2112 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  2304 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit  2400 Meta  type 7F length 7  43 7B
01 45 00 7F 7F
                                        Trk 14       Unit  2496 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit  2592 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  2688 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit  2976 Meta  type 7F length 7  43 7B
```

```
01 45 00 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 33 00 7F 7F
01 41 08 7F 7F
01 33 00 7F 7F
01 45 00 7F 7F
01 33 00 7F 7F
01 45 08 7F 7F
01 33 00 7F 7F
01 45 00 7F 7F
01 37 00 7F 7F
01 41 0A 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 45 00 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 37 00 7F 7F
01 41 08 7F 7F
01 45 08 7F 7F
01 45 00 7F 7F
01 45 08 7F 7F
```

```
Trk 14       Unit  3072 Meta  type 7F length 7  43 7B
Trk 14       Unit  3168 Meta  type 7F length 7  43 7B
Trk 14       Unit  3264 Meta  type 7F length 7  43 7B
Trk 14       Unit  3360 Meta  type 7F length 7  43 7B
Trk 14       Unit  3456 Meta  type 7F length 7  43 7B
Trk 14       Unit  3552 Meta  type 7F length 7  43 7B
Trk 14       Unit  3648 Meta  type 7F length 7  43 7B
Trk 14       Unit  3744 Meta  type 7F length 7  43 7B
Trk 14       Unit  3840 Meta  type 7F length 7  43 7B
Trk 14       Unit  3936 Meta  type 7F length 7  43 7B
Trk 14       Unit  4224 Meta  type 7F length 7  43 7B
Trk 14       Unit  4320 Meta  type 7F length 7  43 7B
Trk 14       Unit  4416 Meta  type 7F length 7  43 7B
Trk 14       Unit  4512 Meta  type 7F length 7  43 7B
Trk 14       Unit  4608 Meta  type 7F length 7  43 7B
Trk 14       Unit  4704 Meta  type 7F length 7  43 7B
Trk 14       Unit  4800 Meta  type 7F length 7  43 7B
Trk 14       Unit  4896 Meta  type 7F length 7  43 7B
Trk 14       Unit  4992 Meta  type 7F length 7  43 7B
Trk 14       Unit  5088 Meta  type 7F length 7  43 7B
Trk 14       Unit  5184 Meta  type 7F length 7  43 7B
Trk 14       Unit  5280 Meta  type 7F length 7  43 7B
Trk 14       Unit  5376 Meta  type 7F length 7  43 7B
Trk 14       Unit  5472 Meta  type 7F length 7  43 7B
Trk 14       Unit  5568 Meta  type 7F length 7  43 7B
Trk 14       Unit  5664 Meta  type 7F length 7  43 7B
Trk 14       Unit  5856 Meta  type 7F length 7  43 7B
Trk 14       Unit  5952 Meta  type 7F length 7  43 7B
Trk 14       Unit  6048 Meta  type 7F length 7  43 7B
Trk 14       Unit  6144 Meta  type 7F length 7  43 7B
Trk 14       Unit  6240 Meta  type 7F length 7  43 7B
Trk 14       Unit  6336 Meta  type 7F length 7  43 7B
Trk 14       Unit  6432 Meta  type 7F length 7  43 7B
Trk 14       Unit  6528 Meta  type 7F length 7  43 7B
Trk 14       Unit  6912 Meta  type 7F length 7  43 7B
Trk 14       Unit  7008 Meta  type 7F length 7  43 7B
Trk 14       Unit  7104 Meta  type 7F length 7  43 7B
```

```
Trk 14       Unit  7200 Meta  type 7F length 7  43 7B 01 41 08 7F 7F
                                        Trk 14       Unit  7296 Meta  type 7F length 7  43 7B
01 45 08 7F 7F
                                        Trk 14       Unit  7392 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit  7488 Meta  type 7F length 7  43 7B
01 45 08 7F 7F
                                        Trk 14       Unit  7584 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  7680 Meta  type 7F length 7  43 7B
01 45 08 7F 7F
                                        Trk 14       Unit  7776 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  7872 Meta  type 7F length 7  43 7B
01 33 00 7F 7F
                                        Trk 14       Unit  7968 Meta  type 7F length 7  43 7B
01 41 08 7F 7F
                                        Trk 14       Unit  7968 EndTrk
```

3.104 Compare all MIDI files

[in menu Analyse]

This batch operation searches for similar or identical MIDI files in two folders.
Using GNMIDI Light license batch operations are not available.

Source folders
Instead of choosing one source and one destination folder (as most other batch operations ask for) you must choose two source folders. The MIDI files within these folders are not modified.

Hint: Only this batch operation allows that both folders are identical folder in order to search for similar MIDI files within one folder. In this case both folder path names must be identically, else it will be assumed that they are different folders and that would cause finding many duplicate files.

Duration
Before comparing starts it will collect MIDI files and some MIDI information from both folders. Then it will compare those MIDI files that have similiar information. The duration for this depends on number of MIDI files in the folder (and sub folders). It could take more time.

Result
At end it generates a text file and displays it with notepad editor, which contains the list of identical or similar MIDI files.

Differences
To get the difference in more detail you could compare two MIDI files.

3.105 Rename MIDI files by song titles

The first track titles will be read from each MIDI file of selected folder (and sub folders). The title will be modified to be compatible for filename use and a file extension .mid will be added.

Danger*:* The original filenames will be renamed! Only use this function with copies of your original files.

Hint:
First track title should contain reasonable song title that the generated file names will help to find your

MIDI songs within archive.

Hint: This operation can not handle unicode characters if used in song title.

Character translation:
Spaces are replaced against underscores _. Except few special characters from German and French language special characters are replaced against _. Some characters that are not allowed in file names are replaced too.

Limit name length for compatibility:
modified title will be cut at maximum length 100 characters.

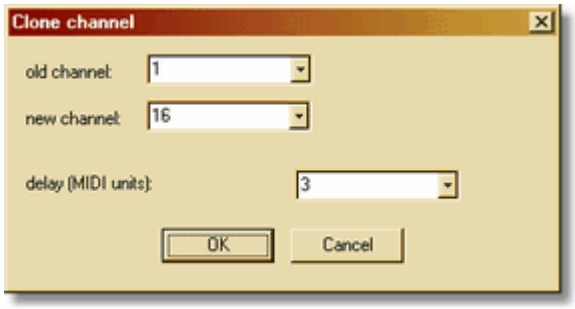
3.106 Rename track names to their MIDI filename

The file names (without path and file extension) of all MIDI files in selected folder (and sub folders) will be set as first track titles.

Danger: The files will be modified! Only use copies of your original files to work with this operation!

Hint: Your file names should contain song name or artist/song name.

3.107 Clone channel



[in menu Modify]

Clone
This operation copies an original channel to a new channel (which should not be free yet) and supports a small delay of the copied MIDI commands. This function will often be used to enhance melody channel by duplication. The delay causes a kind of echo effect.

Old Channel
Der alte Kanal muss in der Originaldatei existieren und MIDI Befehle enthalten. Falls mehrere Spuren diesen Kanal benutzen wird nur aus der Spur mit den meisten Befehlen kopiert.

New Channel
The new channel should not contain MIDI commands before this operation is used. If all channels are already in use then one less important channel needs to be deleted before.

Delay (MIDI units)
All copied MIDI commands can be moved by the specified number of MIDI units back (a pause will be inserted at beginning). Standard value 0 does not cause a delay. Negative values move the commands towards beginning of the song (pauses are removed). In this case there should be an

existing pause at beginning of original channel which can be reduced.

3.108 Clean MIDI song

[in menu Modify/Controller operations]

Clean
This operation removes unnecessary (unused) MIDI commands before first note plays and removes repeated (identical) commands during note area.

Sort
MIDI commands during initial area before first note plays (except drum intro notes) are sorted when they are at same MIDI unit.

Distribute
MIDI commands during initial area before first note plays are distributed by one midi unit and different unit ranges for each channel
(if there is enough time till first note available) so that the number of initial MIDI commands that occur at same time are reduced.

You might turn on sort/distribute completely or partly using following settings in GNMIDI.INI file:

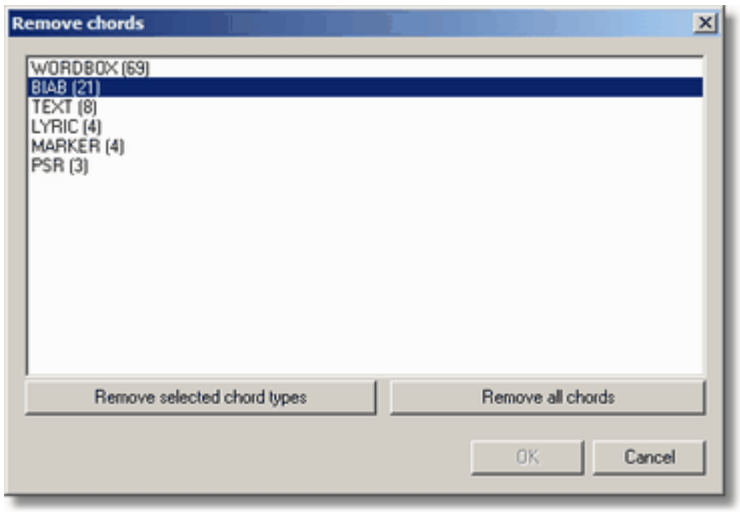
```
[Settings]
MidiCleanAllowSort=1
MidiCleanAllowModifyTime=1
MidiCleanAllowDistributeInitTime=1
```

Hint:
Some critical MIDI commands are not moved or removed (e.g. system exclusive, data controller, all controls off, RPN controller).

Hint:
This operation is only usable for not more than 16 MIDI channels. If your MIDI device supports more than 16 channels and your MIDI song uses them then you should not use this function.

Hint:
Check if the resulting file sounds identical to your original file.

3.109 Remove chords



[in menu Modify]

This function removes chords of some common chord formats used in MIDI files:

- text, lyric, marker chords with following formatting: [Am] (Am) {Am} "Am" %Am DO RE SOL
- PSR Meta chords
- BIAB chords
- Wordbox sysex chords

Select one or more chord formats in the list box and then use remove selected chord types or use Remove all chords.
Use OK button to remove the selected chords or use cancel to abort this operation.

Removing chords might be necessary if you need to add chords in a certain format and the song already contains chords in an other format.

Hint: This operation is available as batch operation. Using GNMIDI Light license batch operations are not available.

3.110 Convert MIDI to (.csv) speadsheet

[in menu Convert]

This converter generates a table in format CSV that contains the MIDI content of a MIDI file (for viewing and modifying with external applications like Microsoft Excel, Microsoft Works, OpenOffice.org ...).

separator: comma (,) or semicolon (;)
character set: ASCII
special characters: are converted into HTML entities (e.g. ).

Hint: The separator can be defined in GNMIDI .ini file

```
[Settings]
CSVColumnSeparator=;
```

Settings: in menu settings you can decide if related NoteOn/NoteOff pairs should be combined to a

Note line.

Hint: Since version 2.56 the .csv table contains new columns: **Time** (for all commands) and **Duration** (for Note, NoteOn commands), which present the position and note duration in milliseconds, **NoteNumber** (for Note, NoteOn, NoteOff commands), which show the MIDI note number corresponding to the note name.

Hint: The .csv file can be modified and may be converted back to a MIDI file.

Hint: If during modification of the table a value in column **Unit** will be removed then the conversion to MIDI file will use the value from column **Time** of same line for the positon calculation.

Hint: If during modification of the table a value from a line Note Zeile the value in column **Length** will be removed then the conversion to MIDI file will use the value from column **Duration** of this line for the note length length calculation.

Hint: If during modification of the table the entry in column NoteName is missing or invalid then the number 0-127 from column NoteNumber will be used for Note, NoteOn, NoteOff commands.

Hint: by default the note names are named sequentially 0... C0 (so that middle C 60 is named C5). The octave number of middle octave could be changed with a gnmidi.ini setting e.g. if you prefer middle C to be C4.
In this case the deepest notes might use a negative octave number (e.g. 0 = C-1).

```
[Settings]
MidiMiddleOctave=5
```

3.110.1 .CSV table format

The csv text (**comma separated text**) uses following fields:
Event,Unit,Time,Track,Channel,NoteName,NoteNumber,NoteUnits,NoteDuration,NoteOnVelocity,NoteOffVelocity,ControllerNr,ControllerValue,TextType,Text,Program,TempoBPM,TempoMicroseconds,Pitchbend,Aftertouch,MeterNom,MeterDenom,MeterUnits,MeterNote32,Resolution,MIDIVersion,Sysex,MetaType,MetaData,Key,PolyKey,PolyValue,RealtimeEvent,RealtimeValue,

They are separated by a csv field separator (usually semicolon or comma).

Each line contains the data of a MIDI command. The lines are sorted by time and sequential order as they occur in the MIDI song.

Event is the name of an event. Using option combined event Note is used else separate events for NoteOn NoteOff are used.

All events contain the fields
Unit (position of the event in MIDI units since start of song)
Time (position of the event in milliseconds since start of song)
Track (track number 1-255, header has no track number)
Channel (channel number 1-16, some events have no channel number like Header, Text, Meta, Sysex, Key ...)

Header is always the first event in the song that uses the fields
MeterNote32 (?)
Resolution (number of MIDI units per beat)
MIDIVersion (0=one track that contains all events sorted by time, 1=several tracks that play parallel, 2=each track is a song)

Text is a META event that contains variable long text of different kind it uses the fields
TextType (name of the text type e.g. trackname, lyric, text, marker, copyright ...)
Text (the text with special characters encoded as HTML literal e.g. #10; for character 10 that is newline)
(warning: song text might not necessarily be stored as lyrics, it could be lyrics, text, marker, META data, sysex and could even be encoded)

Meter is a META event that starts a new bar musical length
MeterNom,MeterDenom (are the musical length of following bars e.g. 3/4)
MeterUnits (?)
MeterNote32 (?)

Key is a META event that describes the musical key of the following part
Key (key name used for displaying by score sheets e.g. F or Fm)

Tempo is a META event that sets a new tempo of the following part
TempoBPM (tempo in beats per minute)
TempoMicroseconds (tempo in microseconds per beat)

META is a general META event of different types
MetaType (number of meta type)
MetaData (a list of values with variable length, values have range 0-127 and are not transferred through MIDI cable)

Sysex is a System exclusive variable length event (has no channel except possibly encoded in its data)
Sysex (a list of hexadecimal values 00-7F, ends with F7)

Realtime are MIDI events like midi clock, stop, continue, start
RealtimeEvent (number of real time event)
RealtimeValue (0-127)

Endtrack is a META event to mark an end of a track

Program is a channel event that sets the program number to be used by the channel
Program (program number 1-128)

Controller is a channel event that sets different channel settings used by the channel
ControllerNr (controller number 0-127)
ControllerValue (controller value 0-127)
(some controllers depend on other controllers set before e.g. RPN, NRPN, data)

NoteOff is a channel event that turns off a note that was playing on the channel
NoteName (name of the note e.g. e4)
NoteNumber (number of the note 0-127)
NoteOffVelocity (note release pressure at end of note 0-127)

NoteOn
NoteName (name of the note e.g. e4)
NoteNumber (number of the note 0-127)
NoteUnits (duration of note in MIDI units)
NoteDuration (duration of note in milliseconds)
NoteOnVelocity (note pressure at starting the note 1-127)

Note when option combine is used contains all NoteOn and NoteOff fields
Hint: the note off position is the note position + duration
Hint: invalid MIDI songs might contain missing start or end

Pitchbend is a channel event that changes pitch of notes up or down on the channel
Pitchbend (0 is middle, negative is down, positive is up, default max. is 2 half tones, range might depend on pitch bend range that some devices can change using rpn or system exclusive messages).

Aftertouch is a channel event that changes pressure of notes that are already playing on the channel
Aftertouch (0-127)

PolyAftertouch is a channel event that changes pressure of certain notes on the channel
PolyKey (note name)
PolyValue (0-127)

3.111 Convert (.csv) spreadsheet to MIDI file

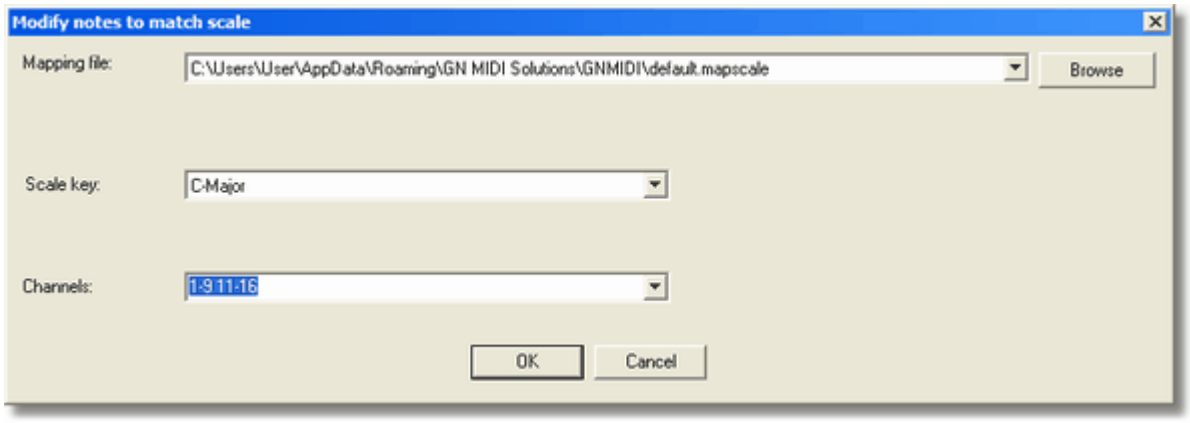
[in menu Convert]

This operation converts tables back to MIDI file that were generated by operation Convert MIDI to (.csv) spreadsheet.

The .csv file can be viewed or modified with a spreadsheet application (e.g. Microsoft Excel, Microsoft Works, OpenOffice.org, LibreOffice)

separator: comma , or semicolon ;
character set: ASCII
special characters: will be treated as characters from Windows ANSI character set
HTML entities: some HTML entities (e.g. ) are converted back to a special character.

3.112 Map notes to scale



[in menu Modify/Note operations]

this operation modifies notes from selected channels to match the selected scale key by mapping rules defined in the chosen scale mapping file.

Mapping file
is a text file with file extension *.mapscale that contains lines in following format:
keyname [notename or notename+increment notename-decrement]
e.g.
C-Major [c dis+1 d dis+1 e f fis+1 g gis+1 a ais+1 bb]

if the chosen scale is not listed or a note is not listed in the line for the scale then the notes are not modified.

Scale key
major and minor scales that can be chosen for the mapping.

Channels
lists the channel numbers (1-16) that chould be modified if necessary.
By default all channels except GM drum channel 10 are selected (1-9 11-16).
Choose a selection from the dropdown list or enter the numbers of channels separated by spaces e.g. 1 3 5 7 9 11 13 15 or e.g. 1-9 10-15

3.113 Use a Play list

[in menu Player]

A **play list** is a text file (*.lst) that contains references to existing songs that should be played in a defined order.
Using GNMIDI Light license play lists are not available.

GNMIDI supports two **play list formats i**n files with file extension *.lst:
simple format: each line contains a path to an MIDI or MP3 song, if the path is relative then it must

be relative to the location of the *.lst file that contains this path. This format can be edited with a simple notepad text editor.
property format: this format is read and generated by application playlist.exe, each lines contains an entry with attributes file="..." filesize="..." title="..."
the title attribute allows to give the entry an alternative title (by default the file name is used). This format will be used if using playlist.exe and changing title attribute for an entry.

Create new play list
use application **playlist.exe** to create a new *.lst file, add some songs or all songs from a folder and optionally sort them manually. Save the play list to a folder where you keep your play lists organised.

Open a play list
choose a play list file with extension *.lst . The application reads the song list and displays errors if the play list contains problems (mostly if referenced songs are not existing).
A new document opens that shows the song entries with numbers in a table. The current selected song has blue text color. If any of the songs in the list is currently playing then its name will be displayed with green text.

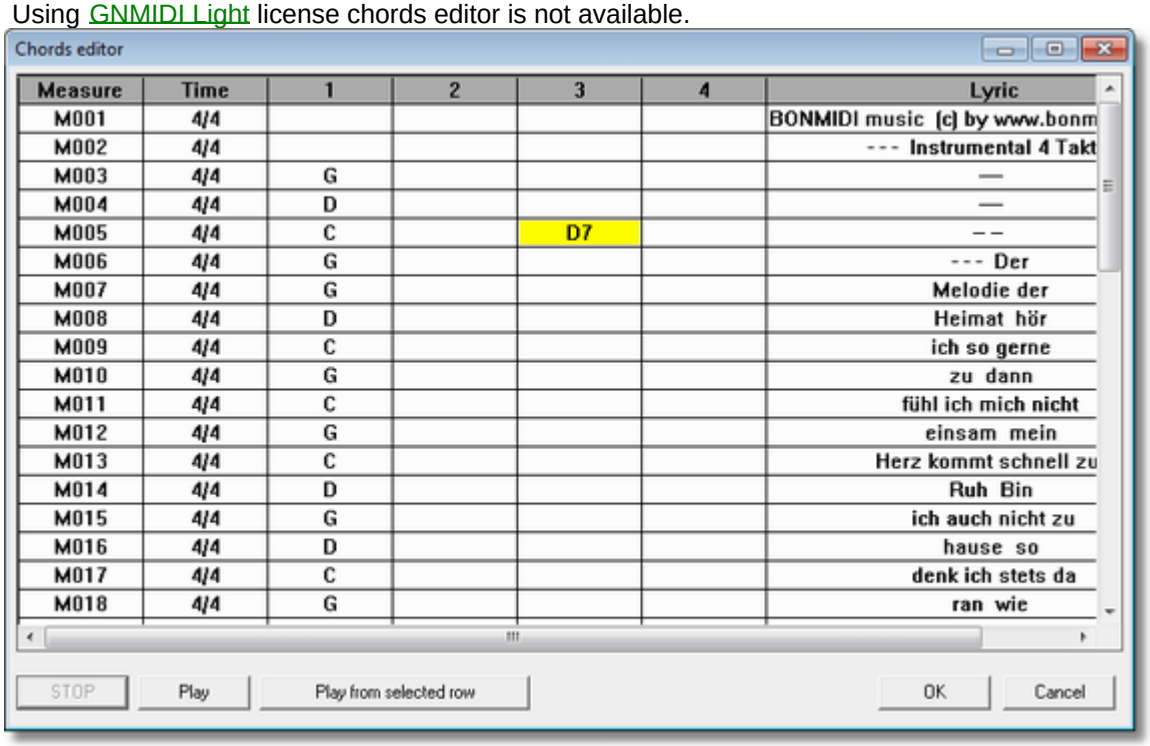
Use the following menu operations to change the selected entry (blue colored):
Select previous song in play list (short key ctrl+arrow left)
Select next song in play list (short key ctrl+arrow right)
Click with right mouse button into a song row to choose this for playing next.

Use the following menu operation to play the next selected song (if current selected song was already playing then this song playing will be stopped and next song behind will be selected and played)
Play next song in play list (short key shift+space)

3.114 Chord editor

[in menu Modify]

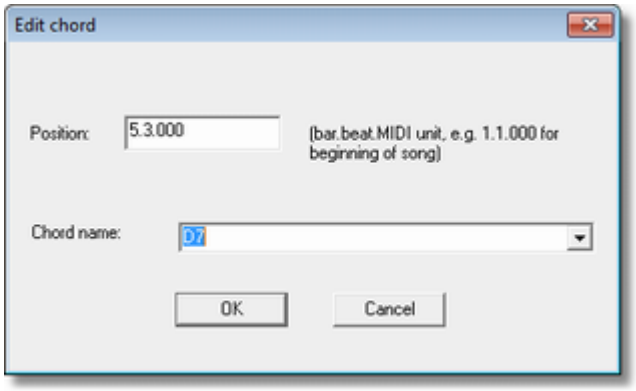
Using GNMIDI Light license chords editor is not available.



The chord editor collects the chords of different formats from the current midi file and displays them in a table. One line for each measure and one beat for each quarternote in the measure. It also displays the lyric words or syllables that begin in a measure.

The chord names in each beat column can be edited. For editing lyrics use the operation edit words.

To **edit a chord name** double click into a table cell. A dialog appears that shows the position and the chord name.



The position can be modified if necessary (normally this is not required). Use the format measure:beat.midiunits (e.g. 11:3.0) where 1:1.0 is the start of the song (first measure, first beat).
The chord name can be entered in the edit text field or if syntax is unclear you could enter a part of the chord name and drop down the list box to **get all possible chord names** that match your input.
Select an entry from the list to use this chord.
Use ok to confirm the new chord or use cancel to abort the dialog.

If a measure has 2/4 and other measures have 4/4 then 4 beat columns will be displayed. For the measure 2/4 only the first 2 beat colums can be used to enter chord.

Deleting a chord or ? symbol (for undetected chords) can be done by double click the cell and removing the chord name in the text edit box.

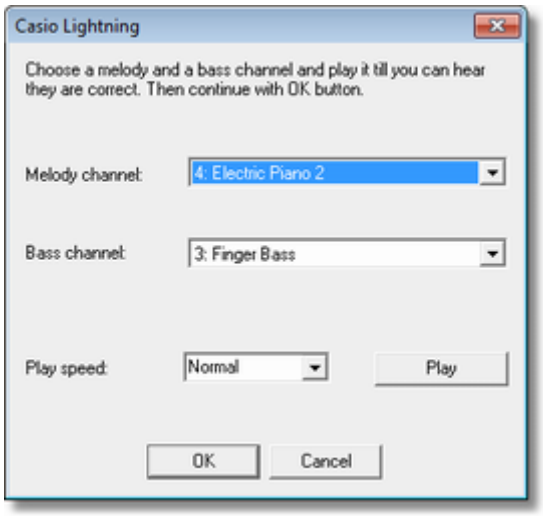
After modifying one or more chord cells use ok button to store the new chords in the format as the original chords had. If the midi song did not contain chords then the chord format lyrics with brackets is used e.g. [Cm]

Storing chords in **BIAB chord** format is not supported. Please use the Band-in-a-box application from PG Music Inc. for modifying BIAB chords.

Play button starts to play the MIDI song beginning from first visible measure.
Play from selected row button starts to play the MIDI song beginning from the row that contains the selected cell.
STOP button stops playing.

While playing the table scrolls the played measures into view and highlights current playing beat.

3.115 Prepare Casio Lightning



[in menu Convert]

Casio has a keyboard serie LK that has the ability to light melody and bass keys while playing. It also helps learning a song since it **lights the key** and waits till using it with left and right hands.
This operation prepares a MIDI file that the melody and bass notes will be lightening with a **Casio LK keyboard.**
If the melody and bass channel are identical then an unused channel is required that the melody and bass notes can be splitted at middle C note.

Melody channel the operation tries to find out the melody channel and might suggest a channel 1-16. You can change the melody channel with the combo box drop down arrow.

Bass channel the operation tries to find out the bass channel and might suggest a channel 1-16. You can change the bass channel with the combo box drop down arrow.

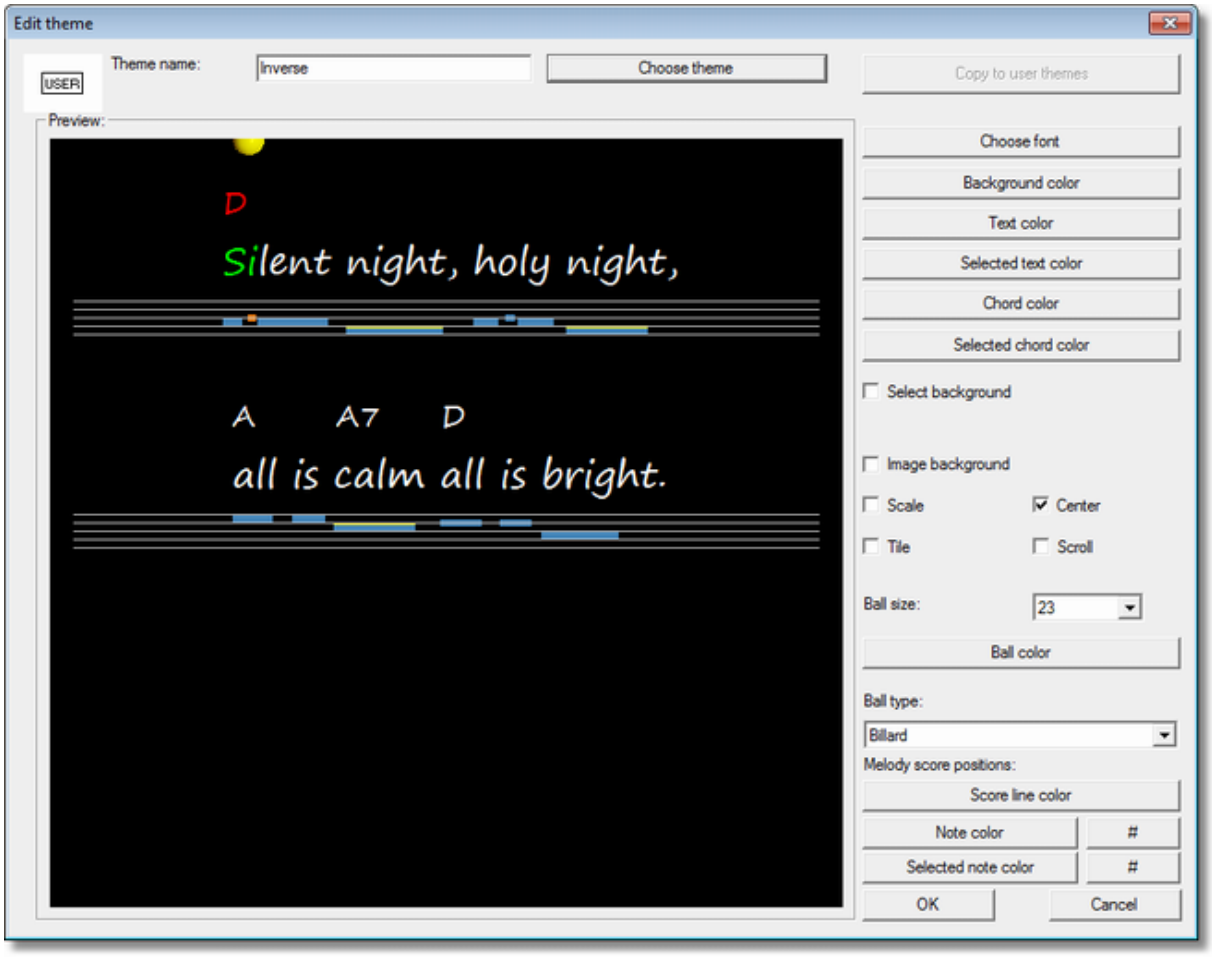
Use **Play bass and melody** button to play a reduced midi song that contains only the melody and bass given above.

Use **OK** button to start the operation. Use **cancel** button to abort the operation.

Hint: In demo version this operation writes only the beginning of a song. Please purchase a GNMIDI license that you can write the whole songs.

Hint: If melody and bass are on same channel this channel must be splitted. For this a free MIDI channel 1-16 is required.
If no free channel is available you will get an error message. Delete a channel that is not so important for the playing.

3.116 Edit theme



`[menu window/Themes...]`

This dialog offers a possibility to choose predefined colors and font for the karaoke view. Such a set of colors and font settings are called a theme. Some standard themes are delivered with the program. This dialog also allows to copy a theme to the user defined themes stored in mygnmidi.ini (in your documents folder) and modify theme according to your favorite taste.

Choose a theme
The button searches for existing themes (in GNMIDI program folder and in your personal documents folder) and displays a popup menu where you can choose a theme with left mouse button click. Click outside of the popup menu to cancel the menu. The chosen theme name will be displayed in the edit box theme name. A preview of the theme settings will be displayed in the preview section.

If the previous loaded theme has been modified or renamed the changes will be stored before loading the selected theme.
A standard theme has icon [STD] and a user defined theme has icon [USER]. A user defined theme can be renamed and modified.

Theme name
The edit box contains the name of current selected theme. For user themes the name can be renamed in this box. Standard theme names can not be renamed. Copy the theme to a user theme to be able to modify or rename the theme.

Copy to user themes
Standard themes can not be modified. This button copies the current loaded theme into file mygnmidi.theme in your personal documents folder. It will automatically get a name beginning with my. The name can be renamed inside the theme name edit box. This button is disabled if the theme is already a user theme in mygnmidi.theme

Choose font
Select a new font that is used for drawing the text. Standardcolors theme uses the last used karaoke font and does not change the font self.

Background color
The background color can be selected with the color picker dialog. Standard theme and standardcolors theme use random color (each document gets a new bright background color).

Text color
Chord color
Selected text color
Selected chord color
These colors and can be selected with the color picker dialog and will be displayed in the preview section. Selected text is used while the playing the song and when this part is about to be played or sung.

Select background
when this option is chosen then the selected chords and selected lyric syllables are drawn with the normal text color but with a background of selected color (default is drawing transparent selected text with selected text color)

Image background
the background can be drawn additionally with an image (only valid JPEG, GIF, BPM on local disk are supported). Turning the checkbox on will open a file dialog to choose an image from your local disk. Uncheck this option to turn off using a background image.

The background image uses some image options:

Scale
the background image is resized to current window size with keeping its aspect ratio. The remaining part at sides is drawn with the background color. (default is drawing with original image size)
Center
the background image is centered into the current window (default is yes, without center it draws the image at top left)

Tile
the background image is drawn multiple as mosaic to fill the current area (default: no).

Scroll
the background image scrolls with the lyrics (default: yes). Otherwise it is drawn at fixed position in the current visible window. The tile option uses this to define where the top left position starts (when fixed it

starts top left in current visible window, when scrolling then it starts at top left of lyrics beginning.
Hint: Without scroll off the drawing of image requires more computer speed that flickering is not visible.

Ball color
the ball type circle uses this color for filling the circle

Ball size
The dance ball size in pixels can be selected and the ball will be redrawn in preview.

Ball type
The dance ball can be a colored circle or a rotating ball (foot ball, billard, water ball). Some balls are already colored and the ball color has no influences.
Hint: Animated balls might consume more processor time. **Do not use animated balls during live concerts**. The ball can be turned on/off in menu settings.

Melody note positions on score lines

Score line color
if background is dark then it might be necessary to choose a bright score line color

Note color, #
normal color of a note position, sharp notes get an other colored line at top

Selected note color,#
selected color of a note position, sharp notes get an other colored line at top

Cancel
When exiting the dialog with cancel button the last modifications will be not stored. A selected theme will not be activated again to be used in karaoke view.

OK
When exiting the dialog with OK button the last modifications (e.g. rename theme name, modified colors, font) will be stored in the user theme and the current selected theme will be activated for use in karaoke view.

Hint: If you already did manual color settings in your gnmidi.ini file then you should keep the color settings for use in a theme definition (in file mygnmidi.ini in documents folder) before you select any new theme.
Choosing a theme will overwrite any previous color settings in your gnmidi.ini file.

Hint: Choosing karaoke font in menu window is deprecated. When using GNMIDI 3.11 or higher you should better use a standard or selfdefined theme instead.

3.117 open demo music file

`[in menu File]`

Gratefully music company bonmidi has provided a demo song with synchronised song text that can be used to try the GNMIDI player and its operations.

Melodie der Heimat - bonmidi music
(GEMA free, demo use only in combination with GNMIDI software, according to written permission by bonmidi Music)

GM MIDI (General Midi version usable on all GM compatible devices)

XG MIDI (Yamaha XG Midi version usable on all Yamaha keyboards or modules)

MP3 with text (recorded song with synchronised song text, 3.6 MB)

melody composer and song text composer: bonmidi music 2017 (all rights reserved)

It is not allowed to distribute the files without written permission of bonmidi music company.

The **complete song** can be purchased in the online MIDI shop

Hint: demo MIDI and MP3 files are not delivered with the GNMIDI program. It requires internet access and http permissions to download the mp3 file once. The file will be downloaded at first time if internet without restrictions is available.

magic flute 14

demonstrates including **pictures** and song **instruction/chord lines** in line by line synchronized midi songs.

The pictures (only small for demonstration only) were added using operation modify/edit words with unit positions calculated by MIDI calculator (from song bar position). Always append <line> after each picture that the picture is displayed on its own line. The absolute path e.g. C:\PICTURES\ was removed in the text line so that the path is automatically taken from the MIDI file location. You can surely use the path if you use a global picture folder on your harddisk (web URLs are not supported). The **syntax <img src="imagepath">** can also be used in synchronisation editor enter lyric dialog.

The instruction lines were added with synchronisation editor directly above the wanted line and lyric character (simply positioned by inserting space characters). A line that contains two or more spaces in a row is assumed to get the same synchronisation time as next line. Append two spaces at end of a line if the instruction or chord line does self not contain SPACESPACE (two spaces). GNMIDI displays the instructions in a line above next lyric line and positions the word close to the expected lyric word in line below (even when non monospaced font is used).

3.118 Export song text into .lrc

(in menu convert)

creates a text file in .lrc format from current midi or mp3 song.

The file name is used from the original song file name and file extension is replaced by .lrc
If a file with this name is already existing in the song folder then a temporary file is created.
The text file is opened by notepad text editor.

GNMIDI and other players search .lrc lyrics in same folder with the same song name (e.g. use golden.lrc for golden.mp3 if existing) if no other lyrics are found inside the song file.

```
[00:00.00]BONMIDI music
[00:00.09](c) by www.bonmidi-music.de (2017) <00:00.10>
[00:00.13]
[00:00.47]Melodie der Heimat <00:00.48>
[00:00.71]Musik und Text: bonmidi music <00:00.71>
[00:00.94]
[00:01.42]Intro Vorzaehler <00:01.42>
[00:01.89]- <00:02.36>- <00:02.83>- <00:02.95>
[00:03.01]Instrumental 4 Takte <00:03.54>
[00:03.78]---- <00:05.67>---- <00:07.56>-- <00:08.50>-- <00:09.15>
[00:09.45]- <00:09.92>- <00:10.39>- <00:10.49>
[00:10.87]Der <00:11.34>Me<00:12.05>lo<00:12.28>die <00:12.76>der
<00:13.23>Hei<00:14.17>mat <00:14.29>
[00:14.65]hör <00:15.12>ich <00:15.83>so <00:16.06>ger<00:16.54>ne <00:17.01>zu
<00:17.60>
[00:18.43]dann <00:18.90>fühl <00:19.61>ich <00:19.84>mich <00:20.31>nicht
<00:20.79>ein<00:21.73>sam <00:21.85>
[00:22.20]mein <00:22.68>Herz <00:23.39>kommt <00:23.62>schnell <00:24.09>zur
<00:24.57>Ruh <00:25.14>
[00:25.98]Bin <00:26.46>ich <00:27.17>auch <00:27.40>nicht
<00:27.87>zu<00:28.35>hau<00:29.29>se <00:29.41>
[00:29.76]so <00:30.24>denk <00:30.94>ich <00:31.18>stets <00:31.65>da<00:32.13>ran
<00:32.72>
[00:33.54]wie <00:34.02>es <00:34.72>so <00:34.96>schön <00:35.43>doch
<00:35.91>wä<00:36.85>re <00:36.97>
[00:37.32]wenn <00:37.80>al<00:38.50>les <00:38.74>ru<00:39.21>hen <00:39.68>kann
<00:40.28>
[00:41.10]Die <00:41.57>Lieb<00:41.81>sten <00:42.05>in <00:42.28>der
<00:42.52>Fer<00:42.99>ne <00:43.46>wis<00:43.94>sen <00:44.06>
[00:44.88]beim <00:45.35>er<00:46.06>sten
<00:46.30>Glock<00:46.77>en<00:47.24>schlag <00:47.83>
[00:48.66]die <00:48.90>Ge<00:49.13>dan<00:49.37>ken <00:49.61>ja <00:49.84>sie
<00:50.08>gehn <00:50.55>auf <00:51.02>Rei<00:51.50>sen <00:51.61>
[00:52.44]und <00:52.91>spie<00:53.62>len <00:53.86>mit <00:54.33>im <00:54.80>Takt
<00:55.39>
[00:56.22]Wir <00:56.69>al<00:56.93>le <00:57.17>sind <00:57.40>so <00:57.64>gern
<00:58.11>bei<00:58.58>sam<00:59.06>men <00:59.17>
[00:59.100]und <01:00.47>war<01:01.18>ten <01:01.42>auf <01:01.89>den <01:02.36>Tag
<01:02.95>
[01:03.78]lasst <01:04.25>al<01:04.49>le <01:04.72>Glock<01:04.96>en <01:05.20>für
<01:05.67>uns <01:06.14>läu<01:06.61>ten <01:06.73>
[01:07.56]zum <01:08.03>Gruß <01:08.74>der <01:08.98>Hei<01:09.45>mat<01:09.92>stadt
<01:10.51>
[01:11.34]Drum <01:11.81>grüß <01:12.52>ich <01:12.76>dich <01:13.23>mein
<01:13.70>Hei<01:14.41>mat<01:14.65>land <01:14.76>
[01:15.12]den <01:15.59>Ort <01:16.30>wo <01:16.54>al<01:17.01>les <01:17.48>liegt
<01:18.07>
[01:18.90]Die <01:19.37>Kind<01:20.08>heit <01:20.31>und <01:20.79>das
<01:21.26>El<01:21.97>tern<01:22.20>haus <01:22.32>
[01:22.68]die <01:23.15>Zeit <01:23.86>liegt <01:24.09>lang
<01:24.57>zu<01:25.04>rück <01:25.64>
```

```
[01:26.46]Die <01:26.93>Jah<01:27.17>re <01:27.40>viel <01:27.64>zu
<01:27.87>schnell <01:28.35>ver<01:28.82>ge<01:29.29>hen <01:29.40>
[01:30.24]vor<01:30.71>bei <01:31.42>der <01:31.65>All<01:32.13>tags<01:32.60>lauf
```

3.119 Export song lyrics and chords into .crd

(in menu convert)

creates a text file in .crd format from current midi or mp3 song.

The file name is used from the original song file name and file extension is replaced by .crd
If a file with this name is already existing in the song folder then a temporary file is created.
The text file is opened by notepad text editor.

lyrics and chord names are only available if the input song contains such information.
The .crd text does not contain synchronization times.

Example:

```
{title: Melodie der Heimat - bonmidi music-xg}
{copyright: (c) by Online-Shop bonmidi music (2017)}
{duration: 1:37}

{key: C}
{time: 4/4}
{tempo: 127}
BONMIDI music
(c) by www.bonmidi-music.de (2017)

Melodie der Heimat
Musik und Text: bonmidi music

Intro Vorzaehler
- - -
Instrumental 4 Takte
---- [G]---- [D]-- [C]-- [D7]
- [G]- -
Der Me[G]lodie der Hei[D]mat
hör ich [C]so gerne zu [G]
dann fühl [C]ich mich nicht ein[G]sam
mein Herz [C]kommt schnell zur Ruh [D]
Bin ich [G]auch nicht zuhau[D]se
so denk [C]ich stets daran [G]
wie es [C]so schön doch wä[G]re
wenn al[D]les ruhen kann
[G]Die Liebsten in [G]der Ferne wissen
[D]beim ersten Glockenschlag
[G]die Gedanken ja sie [G]gehn auf Reisen
[D]und spielen mit im Takt
[G]Wir alle sind so [G]gern beisammen
[D]und warten auf den Tag
[G]lasst alle Glocken für [G]uns läuten
[D]zum Gruß der Heimatstadt
[G]Drum grüß ich dich [D]mein Heimatland
[G]den Ort wo al[D]les liegt
[G]Die Kindheit und [E]das Elternhaus
[A]die Zeit liegt lang [E]zurück
[A]Die Jahre viel zu schnell [G]vergehen
vor[D]bei der Alltagslauf [G]
```

3.120 Check bars

[in menu Analyse]

This operation reads the bar information of a MIDI song and checks if measures are shorter than expected.

Only the last bar is allowed to be shorter because only pauses are assumed to follow.

This operation is available as batch operation in Professional GNMIDI version.

Hint: Bars are not important for playing self but they are useful for printing score sheets and are important for counting (to stay in groove) that no musician starts to play at wrong time.

3.121 Create HTML Listing

[in menu analyse]

The listing operation creates a HTML text or a plain text that contains information about the song content.

For an opened play list it creates a listing of all MIDI and MP3 songs in that play list.

In batch operations (not available in Light license version) the listing operation creates a listing for each folder in a file called gnmidilisting.htm or gnmidilisting.txt

Hint: the operation might take much time if a folder or a play list contains many files.

Hint: if you have a huge collection of songs running the GNMIDI batch tool might take much time. We offer an other product gnmidilisting batch tool that optionally supports caching of song information that speeds up generation of listings for repeated runs.

Hint: GNMIDI uses a fixed layout of the result text outputs. If you need to design your own output text then we offer an other product gnmidilisting batch tool that allows to define and use a template file that describes your output content (for HTML, XML, Text, CSV).

Hint: a second run of the same batch tool will only generate a new listing if a song in this folder has changed or has been added

Hint: If you need to **refresh your result files** then you can use following setting in gnmidi.ini file for the next listing batch operation. It will be set automatically reset back to 0 after the batch listing operation was done.

```
[Settings]
MidiListingRefresh=1
```

Example text listing:

```
--created using GNMIDILISTING v1.0--

1. Melodie der Heimat - bonmidi music-DEMO-gm.mid
    File name: Melodie der Heimat - bonmidi music-DEMO-gm.mid
    File size: 35,5 kb
    Duration: 1:36

    Copyright: (c) by Online-Shop bonmidi music (2017)
    MIDI Format: 0 (single track with multiple channels)
    Resolution: 480  units per beat (1/1920)
    Tempo: 127
    MIDI mode: GM
    Tact changes: 4/4
    Markers: (c) by

Online-Shop bonmidi music

www.bonmidi-music.de
```

info@bonmidi-music.de

26.07.2017

Achtung:

Kein unerlaubtes

Kopieren sowie das

Entfernen des Copyright !!!

Attention:

No unauthorized Copy

and remove the Copyright !!!

Song lyrics:
BONMIDI music
(c) by www.bonmidi-music.de (2017)

Melodie der Heimat
Musik und Text: bonmidi music

Intro Vorzaehler
- - -

Instrumental 4 Takte
---- ---- ----
- - -

Der Melodie der Heimat
hör ich so gerne zu
dann fühl ich mich nicht einsam
mein Herz kommt schnell zur Ruh

Bin ich auch nicht zuhause
so denk ich stets daran
wie es so schön doch wäre
wenn alles ruhen kann

Die Liebsten in der Ferne wissen
beim ersten Glockenschlag
die Gedanken ja sie gehn auf Reisen
und spielen mit im Takt

Wir alle sind so gern beisammen
und warten auf den Tag
lasst alle Glocken für uns läuten
zum Gruß der Heimatstadt

Drum grüß ich dich mein Heimatland
den Ort wo alles liegt
Die Kindheit und das Elternhaus
die Zeit liegt lang zurück

Die Jahre viel zu schnell vergehen
vorbei der Alltagslauf

Tracks: 1

```
Track 1:
        Track title: Melodie der Heimat - bonmidi music-gm
        Channels: 1,2,3,4,5,6,7,8,9,10,11,15,16
        Channel 1:
            Programs: Pan Flute
            Initial volume: 100
            Note count: 72
            Note range: g5-e7
        Channel 2:
            Programs: Finger Bass
            Initial volume: 117
            Note count: 165
            Note range: f#2-d4
        Channel 3:
            Programs: Trumpet
            Initial volume: 115
            Note count: 22
            Note range: d5-e6
        Channel 4:
            Programs: Electric Piano 2
            Initial volume: 117
            Note count: 187
            Note range: a4-d6
        Channel 5:
            Programs: Jazz Guitar
            Initial volume: 93
            Note count: 119
            Note range: a4-e6
        Channel 6:
            Programs: Trumpet
            Initial volume: 103
            Note count: 75
            Note range: g4-c6
        Channel 7:
            Programs: Nylon Guitar
            Initial volume: 74
            Note count: 951
            Note range: g3-b5
        Channel 8:
            Programs: Ensemble Strings1
            Initial volume: 81
            Note count: 142
            Note range: d4-c7
        Channel 9:
            Programs: Mute Guitar
            Initial volume: 82
            Note count: 517
            Note range: g3-a4
        Channel 10:
            Programs: Drums
            Initial volume: 117
            Note count: 1222
            Note range: c3-a#6
        Channel 11:
            Programs: Tubular Bells
            Initial volume: 102
            Note count: 0
            Note range:
        Channel 15:
            Programs: Glockenspiel
            Initial volume: 110
            Note count: 280
            Note range: d4-g5
        Channel 16:
            Programs: Synth Strings2
            Initial volume: 51
```

```
Note count: 129
            Note range: g#4-a5
```

3.122 Create Text Listing

[in menu analyse]

The listing operation creates a HTML text or a plain text that contains information about the song content.

For an opened play list it creates a listing of all MIDI and MP3 songs in that play list.

In batch operations (not available in Light license version) the listing operation creates a listing for each folder in a file called gnmidilisting.htm or gnmidilisting.txt

Hint: the operation might take much time if a folder or a play list contains many files.

Hint: if you have a huge collection of songs running the GNMIDI batch tool might take much time. We offer an other product gnmidilisting batch tool that optionally supports caching of song information that speeds up generation of listings for repeated runs.

Hint: GNMIDI uses a fixed layout of the result text outputs. If you need to design your own output text then we offer an other product gnmidilisting batch tool that allows to define and use a template file that describes your output content (for HTML, XML, Text, CSV).

Hint: a second run of the same batch tool will only generate a new listing if a song in this folder has changed or has been added

Hint: If you need to **refresh your result files** then you can use following setting in gnmidi.ini file for the next listing batch operation. It will be set automatically reset back to 0 after the batch listing operation was done.

```
[Settings]
MidiListingRefresh=1
```

Example text listing:

```
--created using GNMIDILISTING v1.0--

1. Melodie der Heimat - bonmidi music-DEMO-gm.mid
    File name: Melodie der Heimat - bonmidi music-DEMO-gm.mid
    File size: 35,5 kb
    Duration: 1:36

    Copyright: (c) by Online-Shop bonmidi music (2017)
    MIDI Format: 0 (single track with multiple channels)
    Resolution: 480  units per beat (1/1920)
    Tempo: 127
    MIDI mode: GM
    Tact changes: 4/4
    Markers: (c) by

Online-Shop bonmidi music

www.bonmidi-music.de

info@bonmidi-music.de
```

26.07.2017

Achtung:

Kein unerlaubtes

Kopieren sowie das

Entfernen des Copyright !!!

Attention:

No unauthorized Copy

and remove the Copyright !!!

Song lyrics:
BONMIDI music
(c) by www.bonmidi-music.de (2017)

Melodie der Heimat
Musik und Text: bonmidi music

Intro Vorzaehler
- - -

Instrumental 4 Takte
---- ---- ----
- - -

Der Melodie der Heimat
hör ich so gerne zu
dann fühl ich mich nicht einsam
mein Herz kommt schnell zur Ruh

Bin ich auch nicht zuhause
so denk ich stets daran
wie es so schön doch wäre
wenn alles ruhen kann

Die Liebsten in der Ferne wissen
beim ersten Glockenschlag
die Gedanken ja sie gehn auf Reisen
und spielen mit im Takt

Wir alle sind so gern beisammen
und warten auf den Tag
lasst alle Glocken für uns läuten
zum Gruß der Heimatstadt

Drum grüß ich dich mein Heimatland
den Ort wo alles liegt
Die Kindheit und das Elternhaus
die Zeit liegt lang zurück

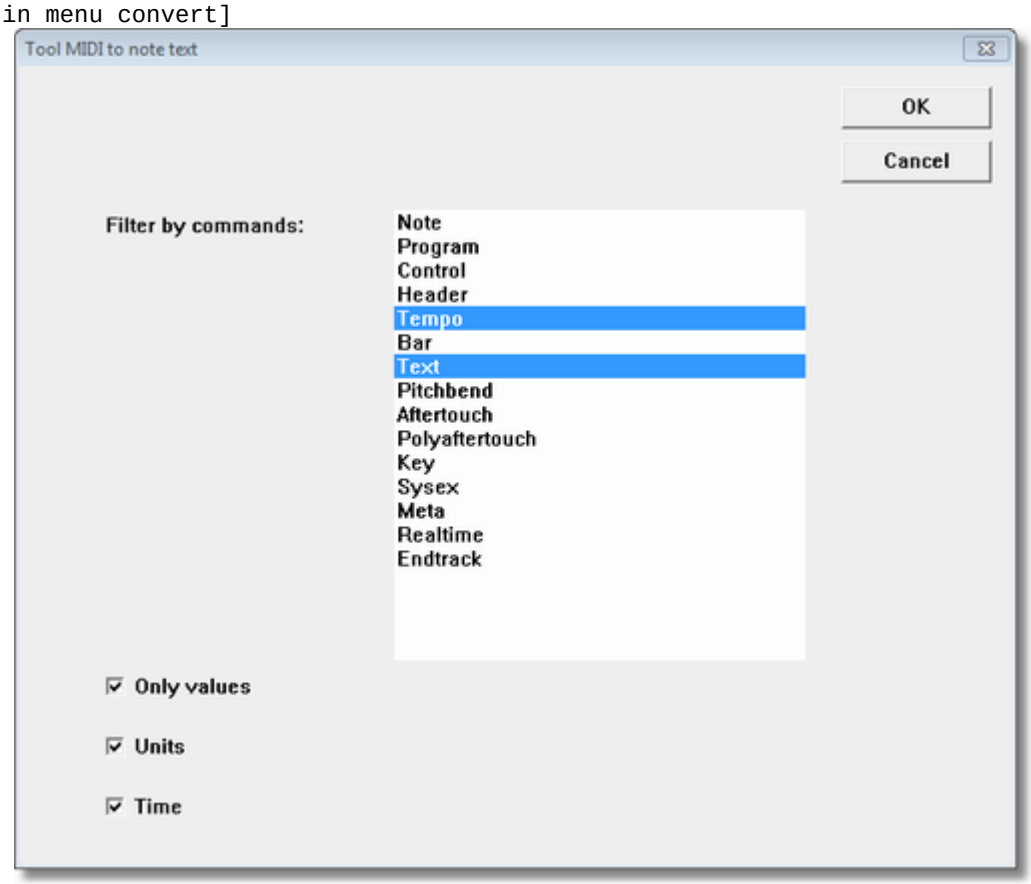
Die Jahre viel zu schnell vergehen
vorbei der Alltagslauf

Tracks: 1
Track 1:
Track title: Melodie der Heimat - bonmidi music-gm

```
Channels: 1,2,3,4,5,6,7,8,9,10,11,15,16
        Channel 1:
            Programs: Pan Flute
            Initial volume: 100
            Note count: 72
            Note range: g5-e7
        Channel 2:
            Programs: Finger Bass
            Initial volume: 117
            Note count: 165
            Note range: f#2-d4
        Channel 3:
            Programs: Trumpet
            Initial volume: 115
            Note count: 22
            Note range: d5-e6
        Channel 4:
            Programs: Electric Piano 2
            Initial volume: 117
            Note count: 187
            Note range: a4-d6
        Channel 5:
            Programs: Jazz Guitar
            Initial volume: 93
            Note count: 119
            Note range: a4-e6
        Channel 6:
            Programs: Trumpet
            Initial volume: 103
            Note count: 75
            Note range: g4-c6
        Channel 7:
            Programs: Nylon Guitar
            Initial volume: 74
            Note count: 951
            Note range: g3-b5
        Channel 8:
            Programs: Ensemble Strings1
            Initial volume: 81
            Note count: 142
            Note range: d4-c7
        Channel 9:
            Programs: Mute Guitar
            Initial volume: 82
            Note count: 517
            Note range: g3-a4
        Channel 10:
            Programs: Drums
            Initial volume: 117
            Note count: 1222
            Note range: c3-a#6
        Channel 11:
            Programs: Tubular Bells
            Initial volume: 102
            Note count: 0
            Note range:
        Channel 15:
            Programs: Glockenspiel
            Initial volume: 110
            Note count: 280
            Note range: d4-g5
        Channel 16:
            Programs: Synth Strings2
            Initial volume: 51
            Note count: 129
            Note range: g#4-a5
```

3.123 User operations

[in menu convert]



example tool with options (this converts MIDI to easy readable text)

GNMIDI contains many operations for modifying or converting MIDI files. But sometimes a user might have a need for very specific operation which can not be found in GNMIDI application.
Since GNMIDI 3.18 a new feature is available where users may solve this lack by user functions.
A user with certain operation wishes may contact info@gnmidi.com and ask if such a tool can be developed.
If possible then the user might order tools that do the wanted tasks. The license price depends on the required work time and
if the user needs the results for commercial uses (company or selling results). Existing standard tools are cheaper.
The tools are delivered with an application that can be used for starting the tools and a license that needs to be installed before the tools can be used.
Some user function might use options like channel numbers or conditions which modifications should be done. These options are displayed in a dialog before conversion starts.
The conversion can be done for a single input file (when the input document is open and active) or as batch convertion of an input folder to an output folder (in batch menu and convert menu when no document is open).

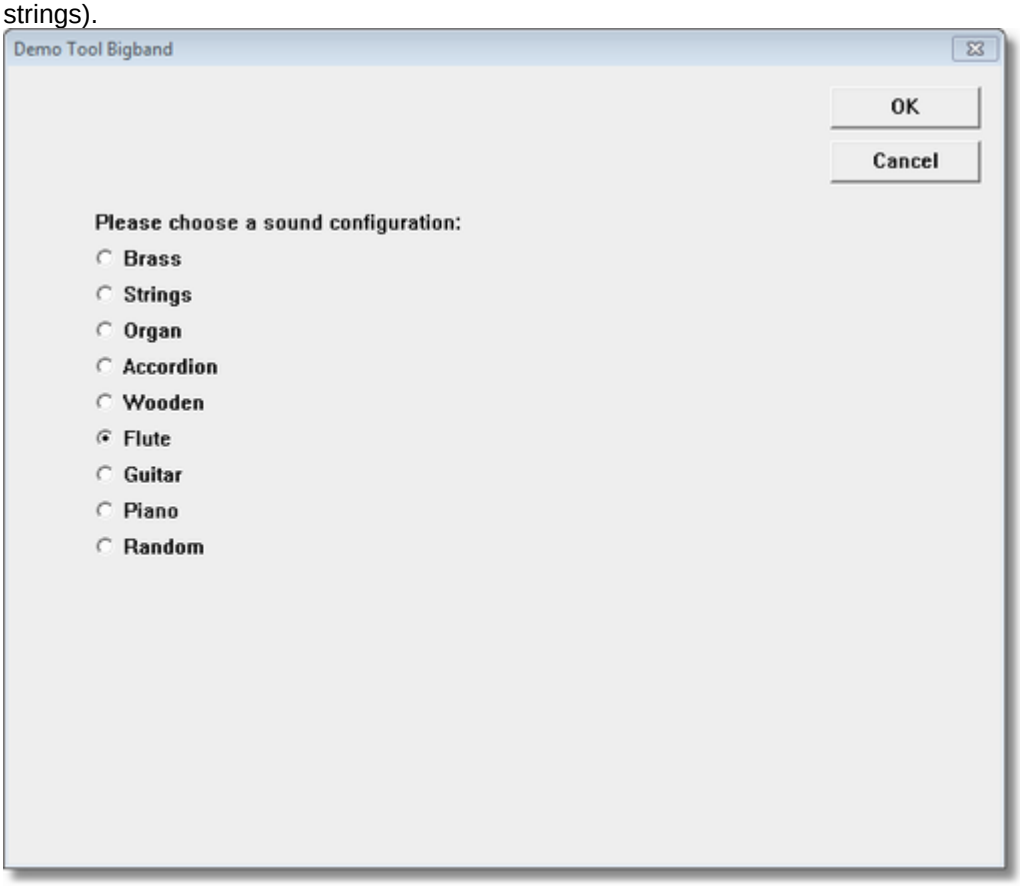
Before using a new tool the license for the tool must be installed and an unused user tool menu item must be clicked to choose the delivered .gntool file using a file dialog.

Converters usually modify MIDI files but could also convert MIDI to an other format (e.g. as the standard MIDI to note text tool does).
A user tool can also convert a text file (e.g. as the standard Note text to MIDI tool does).

Converters usually read input files from a location and write the result to an other location (e.g. convert input folder to output folder).
In some cases exceptions from this rule can be done when the input files and output files are not conflicting (e.g. other file extension) or
if the risk is taken by user if input files are modified directly without generating a second file. The user is responsible for a safe work. It is recommended to backup input files before conversions and working only with copies.

demo tool bigband

this tool demonstrates a user tool.
It changes the sounds of a given GM compatible MIDI song to a chosen sound configuration (e.g. strings).



Programmers who would like to program their own user operations would need a license for gnbatchdialog tool that the own tools are accepted by GNMIDI. It requires knowledge in programming Windows DLL and writing a .gntool and .ini text file that describes how the tool is to use. gnbatchdialog macros are used to build command lines. For manipulating or converting MIDI files it requires knowledge about MIDI file format.

3.124 Set controller values

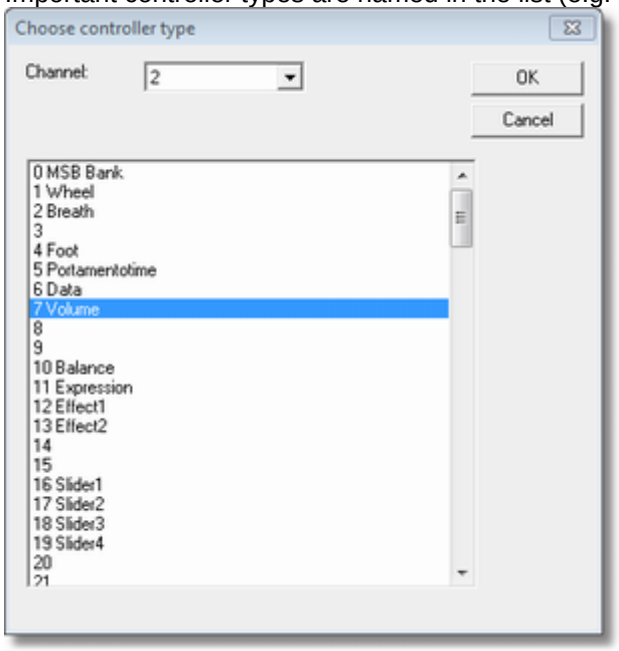
[in menu Modify/controller operations]

A controller command consists of a controller type (0-127) and a controller value (0-127).
Each channel (1-16) has its own controller settings.

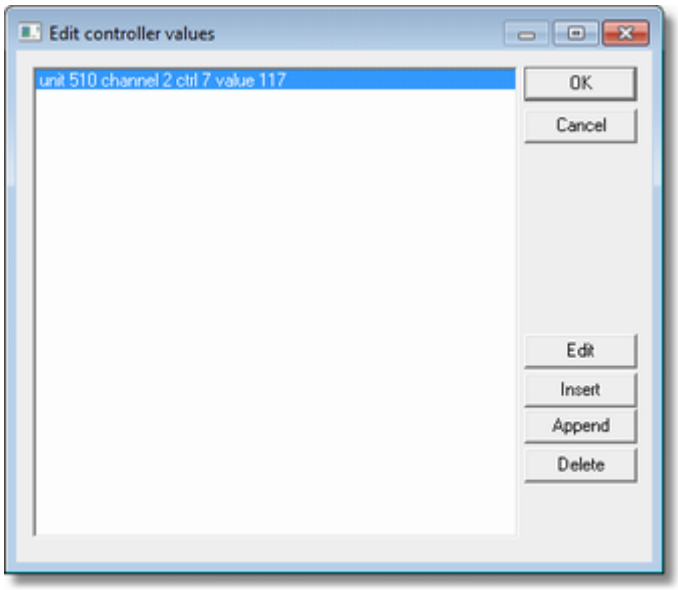
Typical controllers are Volume, balance (panpot), Chorus, Reverb.
These controller values from beginning of a song are displayed in the track list of information window and can be changed, deleted or its position moved with a click into a table cell.

Some **special controller commands** only work in combination with other controllers e.g. Data, RPN, NRPN, MSB Bank, LSB Bank

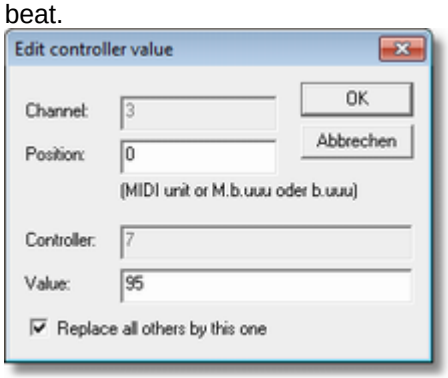
Choose a controller type
First a channel number (1-16) or all channels 1-16 and a controller type (0-127) must be chosen.
Important controller types are named in the list (e.g. 7 Volume).



Edit controller list
For the chosen channel and controller type a list of currently existing controller values and their position inside the MIDI song are displayed.
Select a line to start an operation with this entry.



Edit
with double mouse click or button click a dialog displays that allows to change the controller value and optionally also the position. The controller value must be a number 0 till 127.
The position is a MIDI unit (number 0 or higher) or a beatnumber.unit (b.uuu) or a measure number.beatnumber.unit (M.b.uuu).
Example: 527 2.047 1.2.047 are same position in a MIDI song with note resolution 480 units per beat.



Replace all others by this one
if this option is checked then controllers in same channel of same type are removed and the modified controller remains

Insert
inserts a new controller command before the current controller command (initially with the same position)

Append
inserts a new controller command after the current controller command

Hint: Insert or Append will ask for a channel number for the new controller command

Delete
marks the controller command for deleting. After using OK the controller command will be deleted from the song.

Hint: A controller command will be moved when the position will be changed during Edit.

3.125 Show MIDI initialisation

[in menu analyse]

Before a MIDI song plays notes the song initialises all settings so that the notes will play with correct sound, volume, panpot, effects, pitch etc.
This is often done in a setup tact. Some commands need more pause time till the next command should start since they have more work to do like e.g. Reset GM, XG, GS, GM2, all notes off, all sound off, all controls off
Without enough pause following such a command the effects to direct following commands could be bad for the song. It is often noticed by a track playing default piano instead of the sound chosen.

This command **shows the content of the MIDI tracks** till first note starts.

The result text shows data for each track and channel that uses important commands in the song. If the song track contains commands without channel (e.g. sysex, META) this data is shown before channels data blocks.

Hint: in modify menu the operation edit MIDI initialisation can be used to modify this text and changes are applied to the MIDI song.

Hint: the syntax of this text can be found in page to operation edit MIDI initialisation

3.126 Edit MIDI initialisation

in menu Modify

the operation reads current MIDI file and displays its initialising MIDI commands before first note in a dialog.
The text can be modified and with correct syntax and using OK button the changes can be converted back into the MIDI song and opened as MIDI result.

Each block of commands starts with a header that describes the track number and channel (1-16 or no channel). It is optionally followed by a comment in (...) that may contain track title if available.

```
Track 1 no channel (Melodie der Heimat - bonmidi music-xg):

Track 1 Channel 1 (Melodie der Heimat - bonmidi music-xg):
```

The block optionally ends with === Note starts or === end of track but automatically also ends with beginning of a new track header or end of text.

```
2.1.000 ==== FIRST NOTE
```

Each command has following information inside a text line:
command position in format Measure.beat.tick (where 1.1.000 is the beginning of the song).
command type as text e.g. Control
command parameters e.g. Cxx Vxx for control number xx and control value xx for a Control command
optional comment that describes meaning of a certain control or certain sysex (e.g. (volume))

Following **command types** are available:

Meta FF .. (hexadecimal values except FF follow) e.g. Meta FF 21 00 00 00 00
Sysex F0 ... F7 (hexadecimal values in range 00 - 7F are between) e.g. Sysex F0 7E 7F 09 01 F7 (GM Reset)
Program xx (xx is decimal 1-128) e.g. Program 77 (Shakhchi)
Control Cxx Vxx (xx are values 0-127) Control C0 V0 (bank MSB)

Pitch xx (xx is a value between -8191 and +8191 where 0 means no pitch bending) e.g. Pitch 0

```
Track 1 no channel (Melodie der Heimat - bonmidi music-xg):
1.1.000 Meta FF 00
1.1.000 Meta FF 21 00 00 00 00
1.1.000 Meta FF 43 73 0A 00 04 01
1.1.000 Meta FF 43 7B 00 58 46 30 31 00 11
1.1.000 Sysex F0 7E 7F 09 01 F7 (GM Reset)
1.1.200 Sysex F0 43 10 4C 00 00 7E 00 F7 (XG Reset)
1.2.153 Sysex F0 43 10 4C 02 01 40 06 00 F7
1.2.158 Sysex F0 43 10 4C 02 01 42 11 7C F7
1.2.163 Sysex F0 43 10 4C 02 01 44 1C 74 F7
1.2.168 Sysex F0 43 10 4C 02 01 46 1B 68 F7
1.2.173 Sysex F0 43 10 4C 02 01 48 1B 68 F7
1.2.178 Sysex F0 43 10 4C 02 01 4A 00 4C F7
1.2.183 Sysex F0 43 10 4C 02 01 5A 01 F7
1.2.188 Sysex F0 43 10 4C 04 00 00 59 00 F7
1.2.193 Sysex F0 43 10 4C 02 01 40 01 00 F7
1.2.198 Sysex F0 43 10 4C 04 00 14 0E F7
1.2.203 Sysex F0 43 10 4C 02 01 20 42 08 F7
1.2.208 Sysex F0 43 10 4C 08 09 07 02 F7
2.1.000 ==== FIRST NOTE

Track 1 Channel 1 (Melodie der Heimat - bonmidi music-xg):
1.2.005 Control C0 V0 (bank MSB)
1.2.006 Control C32 V0 (bank LSB)
1.2.007 Program 77 (Shakhchi)
1.2.008 Control C1 V0 (wheel)
1.2.009 Control C7 V105 (volume)
1.2.010 Control C10 V97 (balance)
1.2.011 Control C11 V127 (expression)
1.2.012 Control C64 V0 (hold)
1.2.013 Control C71 V76 (resonance)
1.2.014 Control C72 V67 (sound_release_time)
1.2.015 Control C73 V64 (sound_attack_time)
1.2.016 Control C74 V70 (xg_brightness)
1.2.152 Pitch 0
1.2.208 Control C91 V93 (reverb)
1.2.209 Control C93 V44 (chorus)
1.2.210 Control C94 V32 (xg_effect4)
4.4.260 ==== FIRST NOTE
```

Hint: META text (e.g. lyrics, tracknames), tempo, tact, key, realtime, aftertouch commands are not supported here

3.127 Show MIDI event statistic

[in menu Analyse]
Creates a HTML page containing two tables with statistics of used MIDI events in the song.

First table contains channel events (e.g. notes, programs, controllers). Second table contains events without channel (e.g. text, sysex).

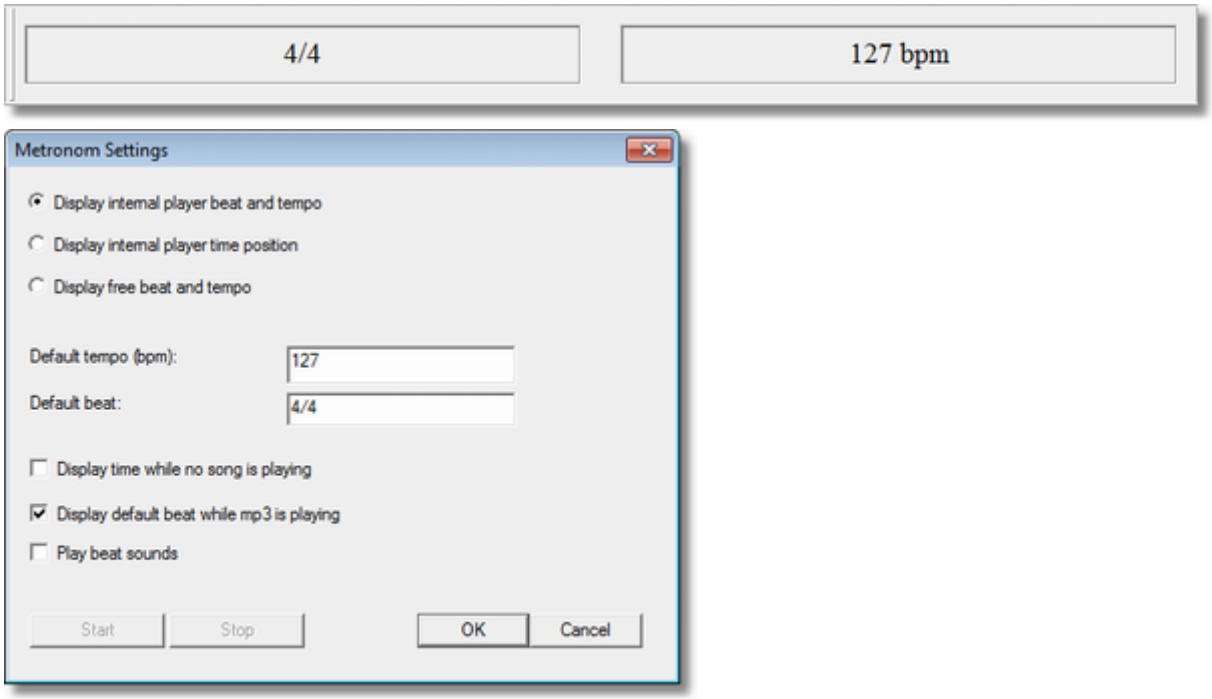
Hint: This operation requires a browser application that opens and displays .htm files by Windows system (e.g. Edge, Chrome, Firefox, ...).

3.128 Show MIDI event parameter statistic

[in menu Analyse]
Creates a HTML page containing 3 tables with statistics of event parameter values used by MIDI events in the song.

First table contains note ranges and note velocity ranges by channel.
Second table contains control value ranges by control and channel.
Third table contains tempo value range (in BPM).
Hint: This operation requires a browser application that opens and displays .htm files by Windows system (e.g. Edge, Chrome, Firefox, ...).

3.129 Metronome



[settings in menu Settings and toggle metronome toolbar visibility in menu window]

The metronome settings dialog is used to define kind and options of a metronome.
In the window menu the metronome toolbar can be displayed or hided. It is only active when the toolbar is visible.

Metronome types

Display internal player beat and tempo
when the internal player plays a MIDI song the metronome toolbar shows current beat e.g. 3/4 and tempo (e.g. 120 bpm). when it is playing a mp3 song the toolbar shows the default beat and tempo given by the settings synchronized to the mp3 song if the option is set.
if no song is playing and the option is set it displays local system time.

Display internal player time position
when the internal player plays the current song time is displayed. If no song is playing and the option

is set it displays local system time.

Display free beat and tempo
after using start the default beat and tempo from the settings is used to display beat and tempo. Stop can be used to top the displaying. If echo sound is on then MIDI drums are played to the output midi device.

Options:

Default tempo (bpm)
a tempo number in beats per minute from 40 - 250 can be set as default tempo. The default tempo is used for mp3 songs and free beat and tempo metronome

Default beat
a beat consists of a nominator and denominator number separated by slash character e.g. 3/4. The default beat is used for mp3 songs and free beat and tempo metronome

Display time while no song is playing
local system time is displayed in metronome toolbar while no song is playing or paused.

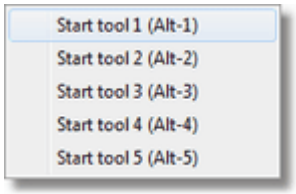
Display default beat while mp3 is playing
Usually mp3 songs do not contain beat information. The default beat can be used to display a synchronized metronome beat to the mp3 song. It is useful if the beat is correctly matching the mp3 rhythm and the beat and tempo does not change during song. The beat is synchronized to the start of song playing.

Play beat sounds
Plays MIDI drum notes at beat positions to the current MIDI output device when this option is set.

Hint: Double left mouse click into middle of the toolbar opens the settings dialog too.

Hint: the toolbar can be docked to upper and lower side of the program.

3.130 Start external tool



[menu_Analyse]

This operation allows to define up external tools that can be started by menu item. First 5 entries can be started by or ALT+number **key shortcut** (e.g. Alt+1, Alt+2 ...).

The commandlines for the application must be set in gnmidi.ini settings:

Example

```
[Settings]
CommandlineExternTool1=%quote(c:\program files(x86)\GN MIDI
Solutions\GNMIXER\gnmixer.exe)%%single( %quotedpath%)%
CaptionExternTool1=GNMixer Software
```

optional variables

`%quotedpath%` will be replaced by the quoted full path of current document in GNMIDI e.g. "c:\midi\Blue line.mid"
Hint: the quotes will be added if necessary (when the path contains spaces, Windows commandline usually requires "..." quotes in this case).

`%unquotedpath%` will be replaced by the full path of current document without any quotes in GNMIDI e.g. c:\midi\Blue line.mid

optional macros

`%quote(text)%`
adds quotes around text if it contains spaces. Special encoding of backslash and end of text and quotes if they are inside the text. cmd.exe requires to quote filenames or folder path so that it is not splitted into words.

`%unquote(text)%`
removes quotes if text starts and ends with quote

`%single(text)%`
when a document is loaded text will be added to the command line. When no document is loaded text will be ignored.

`%nofile(text)%`
when no document is loaded text will be added to the command line. When any document is loaded text will be ignored.

Hint: %single(text)% and %nofile(text)% can be used to create different command lines depending if a document file is available or not (e.g. open application with file path or only open application without document

Hint: use full application path that it can be found.
Hint: the variable can be omitted if the application should be started without any document.
Hint: after first use the operation complains about missing .ini setting and writes example settings into the gnmidi.ini file and opens the gnmidi.ini file with notepad editor.

Hint: at starting applications GNMixer and GNInsertController are added if installed and GNMIDI finds the programs.

3.131 Remove fade in or fade out

`[in menu Modify/volume operations]`
This operation searches for **volume** or **expression** or **note velocities** that are increasing (**fade-in**) or decreasing (**fade-out**) steadily.

criterias:
- at least 5 values in a row must be available
- there must be less than 10 seconds between two values
- there must be at least 2 seconds distance between first and last value
- first value and last value must have a difference of at least 20
- for note velocities the values must change several times
- it is not necessary to fade all channels same time
- the areas do not need to be at beginning or end of song

If no area in song matches the criterias then no changes are made and a message is displayed in status bar.

Else the found areas will be **changed**:
For volume and expression all these controllers except first are removed. The value of the first controller will be set for fade in to the last value and for fade out to the first value.
For note velocities all values will be set to the same value (for fade in to the last value and for fade out to the first value).

The found and modified areas will be listed in the logfile.

3.132 Teleprompt editor

Teleprompt text

Displaying text on a screen helps during reciting text (song text, poem, speech) that no text hangers occur.
For longer text that are displayed over several pages the text on screen must be scrolled.
GNMIDI teleprompt function allows to automatically scroll text by few synchronized positions at appropriate speed.
At unplanned pauses the pause button can be pressed and the presentation continued later at this position.
When situation change very strong (e.g. time for presentation is very different from plans or some passage must be skipped) you can also scroll the text manually from time to time.
A .ptp teleprompt file can be opened with menu File/open and played in karaoke view with Play button.
Other than playing synchronized music pieces (MIDI, MP3) the teleprompt displays no ball and the current line is never colored (assuming that the times are not precise).

Play teleprompter

open a .tpt file with menu File/Open (choose .tpt or *.* in file extension combo box) and use the Play button.
Use pause button to pause the teleprompter and press it again to continue.
Use stop button to end the teleprompter.
The teleprompter playing ends automatically when the <END> line is reached.
During playing the text automatically scrolls according to the calculated speed defined by given time stamps.
The automatic scrolling stops when using manual scrolling by keys, scroll bar or wheel mouse.

New teleprompt

a new .ptp file will be created. The text lines and total duration are required.

Export song text into .lrc

creates a .lrc text file with synchronized telelprompter text lines

Teleprompt editor

a .ptp teleprompt file will be loaded and displayed in the dialog for editing.
You can modify the total duration, the background music and the text lines.

Total duration

the most important time position is the end of the presentation. All other text line times can be calculated assuming a constant speed of presentation.
The duration can be specified in format m:ss (minutes and seconds) or h:mm:ss (hours, minutes, seconds) or
mm:ss.milliseconds (minutes, seconds, milliseconds) . By default the duration 60 seconds will be assumed that can be changed.

Text lines

The text lines should be entered into the edit box.
The lines should not be too large so that the line is visible on screen without horizontal scrolling when using an appropriate font size.
A time stamp can be put at beginning of any line. When editing an already synchronized teleprompt text some synchronized lines will automatically contain the time stamps.

```
[m:ss.milliseconds] text...
[m:ss] text...

[h:mm:ss] text...
```

Silent

Optional a background music (MIDI or MP3) can be played.
By default the teleprompt text is silent.

Background music

use button ... to choose a music piece (.mid or .mp3) from your hard disk.
The Path of this music file will be displayed in the edit field.

3.133 Teleprompt synchronization

A GNMIDI teleprompt is a multiline text where some lines use time stamps to optimize the scrolling speed during displaying.
All unsynchronized lines will automatically get calculated time stamps depending previous and next known time stamp and the amount of text between.
By default the first line gets time stamp 0:00 and duration of the presentation will be the time stamp of end. All time stamps lines between are calculated from the average speed between.

During playing the current play time is displayed and the current line with time between (calculated) time stamp of current line and before (calculated) time stamp of next line is colored red.
Calculated time stamps are displayed with gray text color other than time stamps that are set by user.
Similar to synchronization editor the dialog supports joined lines that are displayed above next line when the text line contains more spaces in series e.g. "The music ".
Such a line automatically gets the same time stamp as the next line. Lines that are joined are usually used for chords, presentation hints (spaces are used to align text above following line).

Play from begin

starts to play teleprompt presentation from beginning. The active line near the (calculated) time stamps are highlighted red.
Background music is playing if the presentation is not silent and the background music file is available and valid MIDI or MP3.
Windows volume slider in taskbar can be used to change volume.

Play before selection

starts to play teleprompt presentation few seconds before the current selected line.

Stop

stops the teleprompt presentation and optional background music.

Edit Time (manually changing a time stamp)

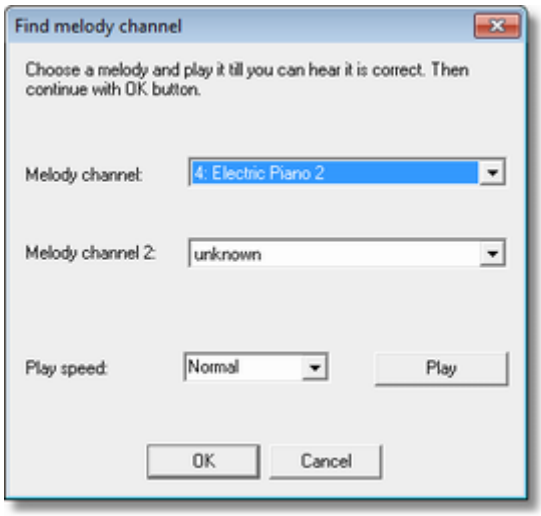
First select a line with mouse click into the list box. Use the button Edit time and enter a time that is between not calculated time stamps before and after this line.
The gray calculated times can be ignored. The time format is minute:second or minute:second.millisecond .
Delete the time in edit box to remove the time stamp for a line (makes it gray again with calculated time stamp)
After pressing OK the valid time will be assigned as time stamp to the current selected line and

joined lines too.
The gray time stamps will be calculated and displayed again.

Synchronize selected line (change time stamps during playing)
while playing the presentation you can select a line with left mouse button and use the button synchronize selected line.
The current play time will be assigned to this line if it is in valid range and also to joined lines.
After successful changing the selected line changes to the next line.

Created teleprompter text can be played with Play button and speed synchronized with the teleprompter synchronization dialog.

3.134 Find melody channel



[menu_analyse]

A MIDI file contains up to 16 MIDI channels. One or more can play the melody.
The melody channel number is not defined by MIDI standard.

E.g. for muting melody or for displaying melody notes on scoreline you must find out the melody channel. This function should help during this task by showing information about MIDI channels (track names and used sound name) and by playing one or two selected MIDI channels while longer pauses are skipped and playing speed can be increased.

Melody channel
Choose a channel from the combo list box which might be possible a melody channel according to description (track name, sound name).

If there is already a melody channel stored for this song (MIDI description) this channel will be suggested. It will try to detect and suggest a melody channel from the track names and similiarities between note positions and lyric syllable positions.

channel 2
Here you can choose a second channel. This allows to quicker analyse if a melody channel is found. Some songs use melody channels.

Hint: When searching melody and bass then channel 2 asks for the **bass channel**.
Hint: channel 10 is reserved for GM drums.

Play speed

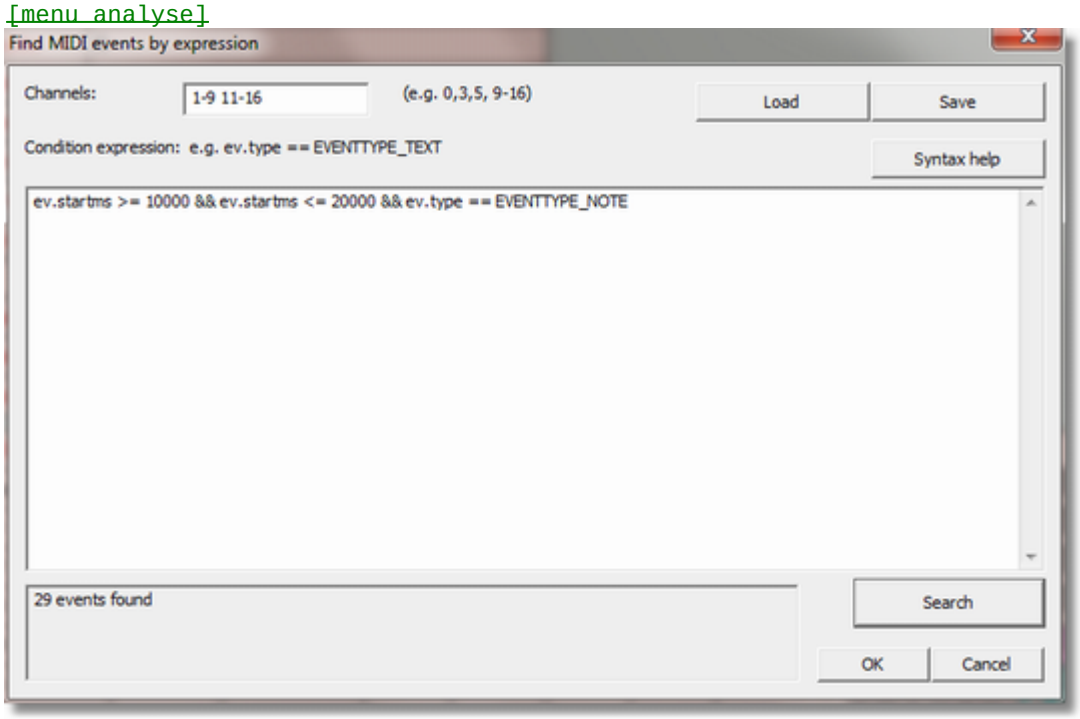
Normal uses the original speed of the MIDI song.
Choose a higher number to play the selected channels quicker if you still can identify the melody.

Play
Plays the selected channels with selected speed.
The playing stops automatically when changing a channel or the speed or when the dialog is closed.

At function "Find melody channel" the chosen channels will be stored as melody channels in the MIDI description to current song after dialog was closed with OK.

3.135 Seek MIDI events by conditions (for experts)

[menu analyse]



Find MIDI events in current MIDI file matching optional channels 0-16 (0=no channel) and a C programming like condition expression. This operation requires some understanding of programming scripts in any language.

channels
Empty field means ignoring any MIDI event channel (same as 0-16).
Number 0 means events that have no MIDI channel (e.g. text, tempo ...).
Numbers 1-16 are MIDI channel numbers (e.g. notes, controllers, sound programs ...)
The field allows to enter a list of numbers between 0 and 16 and also number ranges (e.g. 1,3,5 11-16).
The separators can be spaces or comma.
Only events matching the numbers will be considered by the condition.
Even if a condition may also check channel numbers (e.g. ev.channel in [1,3,5,11..16]) using this field speeds up the duration

condition
This field expects a boolean expression in a C like script language.
See more about MIDI event conditions written in gnscript syntax in page

gnscript condition expressions syntax including examples.

Search
Enter a condition and use button run to start the search.

Syntax errors
errors are displayed in the box below the condition edit box (in English language).

Load
Open a *.dlg text file to load previously stored settings for the dialog

Save
Saves current dialog settings into a .dlg text file for future reuse

Results:
If searching is successful then number of matching events will be displayed in the box below the condition.

OK
closes the dialog and generates an ASCII text file opened in GNMIDI which contains the matching events and some important events for correct MIDI timing (MIDI header, MIDI tempo events, MIDI tact events).

Syntax help
will show this page (also F1 key). Use also gnscript package at https://www.gnmidi.com/gnscript.htm to learn from script examples.

Too difficult
If programming is too difficult for you then we can also offer to program user tools for your special purpose that can be used directly in GNMIDI.
It can analyse or modify or generate MIDI files.
In fact we use in most cases the gnscript language for implementation.

3.135.1 gnscript condition expressions syntax

Few GNMIDI operations use gnscript expressions to match the wanted MIDI events (for experts only). The boolean expression should return 0 (or false) if the current event is not matching and nonzero (or 1 or true) if the event matches.
1 will match all MIDI events.
true will match all MIDI events.
0 (or false) will match no MIDI events.
The current MIDI event information is available as structure variable with name "ev".
The MIDI song information is available as structure with name "song".
The expression can be simple by comparing operators (==, !=, >=, >, <, <=) or
complex with logic operators (&&, ||) or
function calls (e.g. measureatunit(song,ev.startunit) <= 2) and even
calling own function definitions e.g. return ev.everyfifths(); function everyfifths(&ev) { return ev.eventindex() % 5 == 0;
};

gnscript language
https://www.gnmidi.com/gnscript.htm is a demo script interpreter with many script examples for learning programming. It is easier to learn programming by examples than by syntax.
There you can find the complete syntax of the script language.
Hint: gnscript language self does not contain MIDI support. GNMIDI and GNMIDI user tools use gnscript and a MIDI extension that supports MIDI file operations (e.g. variables song, ev, functions like

midiload, midisave ...).

variables

```
    song     is a structure song information
             song.filename     a string
             song.events       a vector of MidiEvent objects

    ev       is a structure of event informations
          ev.type        can be
               EVENTTYPE_NOTE
               EVENTTYPE_PROGRAM
               EVENTTYPE_CONTROL
               EVENTTYPE_HEADER
               EVENTTYPE_TEMPO
               EVENTTYPE_TACT
               EVENTTYPE_TEXT
               EVENTTYPE_PITCHBEND
               EVENTTYPE_AFTERTOUCH
               EVENTTYPE_POLYAFTERTOUCH
               EVENTTYPE_KEY
               EVENTTYPE_SYSEX
               EVENTTYPE_META
               EVENTTYPE_REALTIME
               EVENTTYPE_ENDTRACK

          depending on the event type events have different attributes
          ev.track      track number (1,...)
          ev.channel    channel number (0 for no channel, 1-16)
          ev.startunit  MIDI unit where event started (0 is song start)
          ev.endunit    event EVENTTYPE_NOTE contains MIDI unit where note event
finished  (endunit must be greater or equal startunit)
          ev.length     event EVENTTYPE_NOTE contains note length in MIDI units (can
also be 0)
          ev.startms    start time in milliseconds where event started (0 is song
start)
          ev.endms      event EVENTTYPE_NOTE contains end time in milliseconds where
note event finished
          ev.duration   event EVENTTYPE_NOTE contains note duration in milliseconds
          ev.velocity         event EVENTTYPE_NOTE contains note on velocity (1-127)
          ev.velocityoff     event EVENTTYPE_NOTE contains note off velocity (0-127)
          ev.hasstart   can be 0 when note on/off are not combined
          ev.hasend     can be 0 when note on/off are not combined
          ev.iscombined  1 when note on and off are combined to a pair in the ev
structure
          ev.unitsperbeat   event EVENTTYPE_HEADER contains MIDI resolution as units
per beat
          ev.version    event EVENTTYPE_HEADER contains MIDI version 0, 1, 2
          ev.trackcount   event EVENTTYPE_HEADER contains number of MIDI tracks
          ev.bpm         event EVENTTYPE_TEMPO contains beats per minute tempo
number
          ev.microsecondsperbeat    event EVENTTYPE_TEMPO contains microseconds per
beat  tempo number
          ev.nomin       event EVENTTYPE_TACT contains nominator of tact (e.g. 3 of
3/4)
          ev.denom       event EVENTTYPE_TACT contains denominator of tact (e.g. 4
of 3/4)
          ev.texttype    event EVENTTYPE_TEXT contains text type number
                meta_text
                meta_copyright
                meta_trackname
                meta_instrument
                meta_lyric
                meta_marker
                meta_cuepoint
                meta_programname
                meta_devicename
```

```
ev.program     event EVENTTYPE_PROGRAM contains program number (1,...)
          ev.programname   event EVENTTYPE_PROGRAM contains GM program name (1,...)
          ev.note        event EVENTTYPE_NOTE contains note number (0-127)
          ev.notename    event EVENTTYPE_NOTE contains note name
          ev.controlnr   event EVENTTYPE_CONTROL contains control number (0-127)
                   ev.ctrl_highbank
                   ev.ctrl_wheel
                   ev.ctrl_breath
                   ev.ctrl_foot
                   ev.ctrl_portamentotime
                   ev.ctrl_data
                   ev.ctrl_volume
                   ev.ctrl_balance
                   ev.ctrl_expression
                   ev.ctrl_effect1
                   ev.ctrl_effect2
                   ev.ctrl_slider1
                   ev.ctrl_slider2
                   ev.ctrl_slider3
                   ev.ctrl_slider4
                   ev.ctrl_lowbank
                   ev.ctrl_lowdata
                   ev.ctrl_hold
                   ev.ctrl_portamento
                   ev.ctrl_sustenuto
                   ev.ctrl_soft
                   ev.ctrl_legato
                   ev.ctrl_hold2
                   ev.ctrl_sound_variation
                   ev.ctrl_resonance
                   ev.ctrl_sound_release_time
                   ev.ctrl_sound_attack_time
                   ev.ctrl_xg_brightness
                   ev.ctrl_xg_portamento
                   ev.ctrl_reverb
                   ev.ctrl_tremolo
                   ev.ctrl_chorus
                   ev.ctrl_xg_effect4
                   ev.ctrl_phaser_level
                   ev.ctrl_datainc
                   ev.ctrl_datadec
                   ev.ctrl_lownrpn
                   ev.ctrl_highnrpn
                   ev.ctrl_lowrpn
                   ev.ctrl_highrpn
                   ev.ctrl_allsoundoff
                   ev.ctrl_allcontroloff
                   ev.ctrl_localkeyboard
                   ev.ctrl_allnotesoff
                   ev.ctrl_omnioff
                   ev.ctrl_omnion
                   ev.ctrl_mono
                   ev.ctrl_poly
          ev.controlvalue  event EVENTTYPE_CONTROL contains control value (0-127)
          ev.text        event EVENTTYPE_TEXT contains text
          ev.key
          ev.pitchbend       event EVENTTYPE_PITCHBEND contains pitchbend number 0
is no pitchbend
                   pitch_center = 0
                   pitch_maxdown = -0x2000
                   pitch_maxup = 0x1fff

          ev.aftertouch      event EVENTTYPE_AFTERTOUCH contains aftertouch value
          ev.polykey         event EVENTTYPE_POLYAFTERTOUCH contains polyaftertouch
key
          ev.polyvalue       event EVENTTYPE_POLYAFTERTOUCH contains polyaftertouch
```

```
value
          ev.key              event EVENTTYPE_KEY contains key
          ev.metatype         event EVENTTYPE_META contains meta type value
                meta_seqnumber
                meta_prefixchannel
                meta_prefixport
                meta_endtrack
                meta_tempo
                meta_smpte
                meta_meter
                meta_key
          ev.meta             event EVENTTYPE_META contains meta data as hexadecimal
string (values 00 - 7F)
          ev.metadatalength   event EVENTTYPE_META contains length of meta data in
bytes
          ev.realtimeevent    event EVENTTYPE_REALTIME contains real time event
number
                event_songpos
                event_songselect
                event_tunerequest
                event_clock
                event_start
                event_continue
                event_stop
                event_activesense
          ev.realtimeparam    event EVENTTYPE_REALTIME contains real time event
parameter value
          ev.sysex       event EVENTTYPE_SYSEX contains sysex data as hexadecimal
string (values 00 - 7F or F7)
                sysex_GMReset = "7E 7F 09 01 F7"
                sysex_GMExit = "7E 7F 09 02 F7"
                sysex_GM2Reset = "7E 7F 09 03 F7"
                sysex_GSReset = "41 10 42 12 40 00 7F 00 41 F7"
                sysex_GSExit = "41 10 42 12 40 00 7F 7F 42 F7"
                sysex_XGReset = "43 10 4C 00 00 7E 00 F7"
          ev.sysexdatalength   event EVENTTYPE_SYSEX contains sysex data length in
bytes
```

Hint: constants listed above should be used e.g. ev.type == EVENTTYPE_CONTROL && ev.controlnr == ctrl_volume
Important: many of those event parameters can only be accessed when the current MIDI command has a compatible event type e.g. ev.velocity is only available for notes ev.type == EVENTTYPE_NOTE and else the script aborts with an error. Therefore restrict in the condition the command type e.g. ev.type == EVENTTYPE_NOTE && ev.velocity >= 80

functions (more can be found in gnscript package)

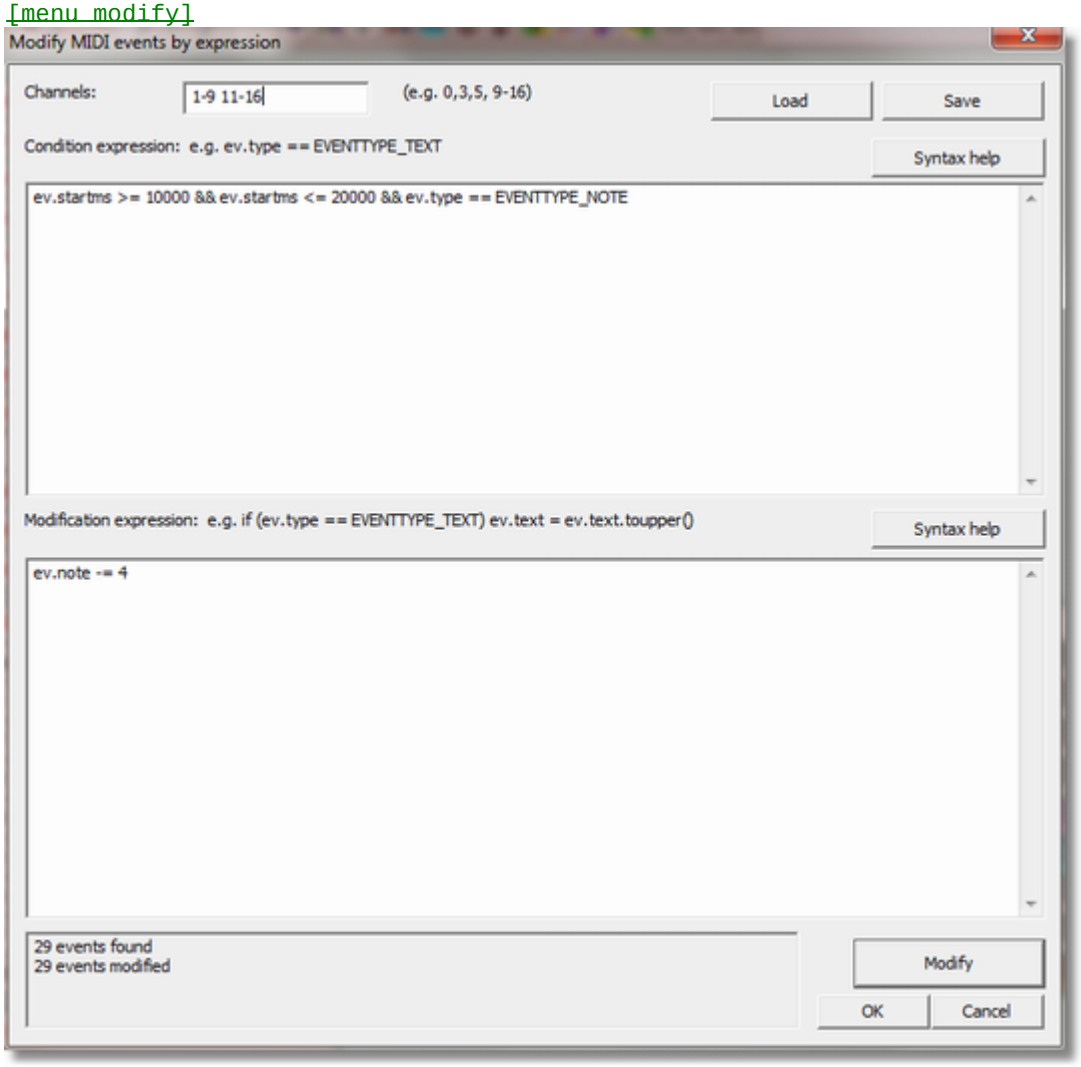
```
  random(maxvalue)      a random value between 0 and maxvalue-1
  text.length()         number of characters in text
  text.index(subtext)   first position of subtext inside text or -1 if not found
  text.toupper()        change all characters a-z to A-Z
  text.tolower()        change all characters A-Z to a-z
  text.mid(pos, len)    get part of text beginning at index pos and maximum length
  len
  text.left(len)        get part of text from beginning with maximum length len
  song.measureatunit(unit)    returns 1 for first measure
  song.lastunit()         returns last used midi unit in song
  song.gettrackcount()    returns number of tracks
  song.getresolution()    returns units per beat
  ev.eventindex()         index of the event inside the song
  ev.ischordevent()       returns true if event contains a known chord
  ev.event2string()      returns a text that contains event information
```

Condition examples:

```
ev.aftertouch == 0
random(10000)==0
true
ev.type == EVENTTYPE_TEXT && ev.texttype == meta_trackname
measureatunit(song,ev.startunit) in  [2..3]
ev.startms > 0 && ev.startms <= 3000
return ev.everyfifths(); function everyfifths(&ev) { return ev.eventindex() % 5 ==
0; };
```

3.136 Modify MIDI events by expression

[menu_modify]



Modify selected MIDI events in current MIDI file matching optional channels 0-16 (0=no channel) and a C programming like condition expression by one or more script commands. This operation requires some understanding of programming scripts in any language.

channels

Empty field means ignoring any MIDI event channel (same as 0-16).
Number 0 means events that have no MIDI channel (e.g. text, tempo ...).
Numbers 1-16 are MIDI channel numbers (e.g. notes, controllers, sound programs ...)
The field allows to enter a list of numbers between 0 and 16 and also number ranges (e.g. 1,3,5 11-

16).
The separators can be spaces or comma.
Only events matching the numbers will be considered by the condition.
Even if a condition may also check channel numbers (e.g. ev.channel in [1,3,5,11..16]) using this field speeds up the duration

condition expression
This optional field expects a boolean expression in a C like script language. By default all events are matching (that also match given channels).
If filled then only matching events are modified by the modification expression.
See more about MIDI event condition expressions written in gnscript syntax in page gnscript condition expressions syntax including examples.
Hint: Use function Seek MIDI events by conditions (for experts) from menu analyse to test the condition expression.

modification expression
This field expects one or more expressions that modify current ev. if or switch might be used to conditionally modify events by certain criterias.
Multiple statements must be separated by semicolon (;).
See more about MIDI event modifiction expressions written in gnscript syntax in page gnscript event modification expression syntax including examples.

Modify
Enter optional channel, optional condition and modification expression and use button run to start the modification.

Syntax errors
errors are displayed in the box below the modification edit box (in English language).

Load
Open a *.dlg text file to load previously stored settings for the dialog

Save
Saves current dialog settings into a .dlg text file for future reuse

Results:
If searching and modification is successful then number of matching events and number of modified events will be displayed in the box below the condition.

OK
closes the dialog and generates a MIDI file opened in GNMIDI which contains the modifications done by the gnscript expressions.

Syntax help
will show this page (also F1 key). Use also gnscript package at https://www.gnmidi.com to learn from script examples.

Batch operation
this operation is also in menu batch operations to modify many files at once. You should first test a script with single files and then with a folder containing few MIDI files before convert a big folder with many files. The batch operation will first ask for the channels/condition script/modification script and then ask for input and output folders and then it can be started. When the script contains a syntax error it aborts at first file and displays the error including source position.

Too difficult

If programming is too difficult for you then we can also offer to program user tools for your special purpose that can be used directly in GNMIDI.
It can analyse or modify or generate MIDI files.
In fact in most cases we can use the gnscript language for implementation.

3.136.1 gnscript event modification expression syntax

Few GNMIDI operations use gnscript expressions to modify MIDI events (for experts only).
The current event can be modified or removed by one or more commands and conditionally.

gnscript language
https://www.gnmidi.com/gnscript.htm is a demo script interpreter with many script examples for learning programming. It is easier to learn programming by examples than by syntax.
There you can find the complete syntax of the script language.
Hint: gnscript language self does not contain MIDI support. GNMIDI and GNMIDI user tools use gnscript and a MIDI extension that supports MIDI file operations (e.g. variables song, ev, functions like midiload, midisave ...).

variables

song is a structure song information

```
song.filename     a string
song.events       a vector of MidiEvent objects
```

ev is a structure of event informations

ev.type can be

```
EVENTTYPE_NOTE
EVENTTYPE_PROGRAM
EVENTTYPE_CONTROL
EVENTTYPE_HEADER
EVENTTYPE_TEMPO
EVENTTYPE_TACT
EVENTTYPE_TEXT
EVENTTYPE_PITCHBEND
EVENTTYPE_AFTERTOUCH
EVENTTYPE_POLYAFTERTOUCH
EVENTTYPE_KEY
EVENTTYPE_SYSEX
EVENTTYPE_META
EVENTTYPE_REALTIME
EVENTTYPE_ENDTRACK
```

depending on the event type events have different attributes

```
ev.track      track number (1,...)
ev.channel    channel number (0 for no channel, 1-16)
ev.startunit  MIDI unit where event started (0 is song start)
ev.endunit    event EVENTTYPE_NOTE contains MIDI unit where note event
finished  (endunit must be greater or equal startunit)
ev.length     event EVENTTYPE_NOTE contains note length in MIDI units (can
also be 0)
ev.startms    start time in milliseconds where event started (0 is song
start)
ev.endms      event EVENTTYPE_NOTE contains end time in milliseconds where
note event finished
ev.duration   event EVENTTYPE_NOTE contains note duration in milliseconds
ev.velocity         event EVENTTYPE_NOTE contains note on velocity (1-127)
ev.velocityoff     event EVENTTYPE_NOTE contains note off velocity (0-127)
ev.hasstart   can be 0 when note on/off are not combined
ev.hasend     can be 0 when note on/off are not combined
ev.iscombined  1 when note on and off are combined to a pair in the ev
structure
ev.unitsperbeat   event EVENTTYPE_HEADER contains MIDI resolution as units
per beat
```

```
ev.version    event EVENTTYPE_HEADER contains MIDI version 0, 1, 2
          ev.trackcount   event EVENTTYPE_HEADER contains number of MIDI tracks
          ev.bpm         event EVENTTYPE_TEMPO contains beats per minute tempo
number
          ev.microsecondsperbeat    event EVENTTYPE_TEMPO contains microseconds per
beat  tempo number
          ev.nomin       event EVENTTYPE_TACT contains nominator of tact (e.g. 3 of
3/4)
          ev.denom       event EVENTTYPE_TACT contains denominator of tact (e.g. 4
of 3/4)
          ev.texttype    event EVENTTYPE_TEXT contains text type number
                meta_text
                meta_copyright
                meta_trackname
                meta_instrument
                meta_lyric
                meta_marker
                meta_cuepoint
                meta_programname
                meta_devicename

          ev.program     event EVENTTYPE_PROGRAM contains program number (1,...)
          ev.programname   event EVENTTYPE_PROGRAM contains GM program name (1,...)
          ev.note        event EVENTTYPE_NOTE contains note number (0-127)
          ev.notename    event EVENTTYPE_NOTE contains note name
          ev.controlnr   event EVENTTYPE_CONTROL contains control number (0-127)
                   ev.ctrl_highbank
                   ev.ctrl_wheel
                   ev.ctrl_breath
                   ev.ctrl_foot
                   ev.ctrl_portamentotime
                   ev.ctrl_data
                   ev.ctrl_volume
                   ev.ctrl_balance
                   ev.ctrl_expression
                   ev.ctrl_effect1
                   ev.ctrl_effect2
                   ev.ctrl_slider1
                   ev.ctrl_slider2
                   ev.ctrl_slider3
                   ev.ctrl_slider4
                   ev.ctrl_lowbank
                   ev.ctrl_lowdata
                   ev.ctrl_hold
                   ev.ctrl_portamento
                   ev.ctrl_sustenuto
                   ev.ctrl_soft
                   ev.ctrl_legato
                   ev.ctrl_hold2
                   ev.ctrl_sound_variation
                   ev.ctrl_resonance
                   ev.ctrl_sound_release_time
                   ev.ctrl_sound_attack_time
                   ev.ctrl_xg_brightness
                   ev.ctrl_xg_portamento
                   ev.ctrl_reverb
                   ev.ctrl_tremolo
                   ev.ctrl_chorus
                   ev.ctrl_xg_effect4
                   ev.ctrl_phaser_level
                   ev.ctrl_datainc
                   ev.ctrl_datadec
                   ev.ctrl_lownrpn
                   ev.ctrl_highnrpn
                   ev.ctrl_lowrpn
                   ev.ctrl_highrpn
```

```
ev.ctrl_allsoundoff
                  ev.ctrl_allcontroloff
                  ev.ctrl_localkeyboard
                  ev.ctrl_allnotesoff
                  ev.ctrl_omnioff
                  ev.ctrl_omnion
                  ev.ctrl_mono
                  ev.ctrl_poly
         ev.controlvalue  event EVENTTYPE_CONTROL contains control value (0-127)
         ev.text        event EVENTTYPE_TEXT contains text
         ev.key
         ev.pitchbend        event EVENTTYPE_PITCHBEND contains pitchbend number 0
is no pitchbend
                 pitch_center = 0
                 pitch_maxdown = -0x2000
                 pitch_maxup = 0x1fff

         ev.aftertouch       event EVENTTYPE_AFTERTOUCH contains aftertouch value
         ev.polykey          event EVENTTYPE_POLYAFTERTOUCH contains polyaftertouch
key
         ev.polyvalue        event EVENTTYPE_POLYAFTERTOUCH contains polyaftertouch
value
         ev.key              event EVENTTYPE_KEY contains key
         ev.metatype         event EVENTTYPE_META contains meta type value
               meta_seqnumber
               meta_prefixchannel
               meta_prefixport
               meta_endtrack
               meta_tempo
               meta_smpte
               meta_meter
               meta_key
         ev.meta            event EVENTTYPE_META contains meta data as hexadecimal
string (values 00 - 7F)
         ev.metadatalength  event EVENTTYPE_META contains length of meta data in
bytes
         ev.realtimeevent    event EVENTTYPE_REALTIME contains real time event
number
               event_songpos
               event_songselect
               event_tunerequest
               event_clock
               event_start
               event_continue
               event_stop
               event_activesense
         ev.realtimeparam    event EVENTTYPE_REALTIME contains real time event
parameter value
         ev.sysex       event EVENTTYPE_SYSEX contains sysex data as hexadecimal
string (values 00 - 7F or F7)
               sysex_GMReset = "7E 7F 09 01 F7"
               sysex_GMExit = "7E 7F 09 02 F7"
               sysex_GM2Reset = "7E 7F 09 03 F7"
               sysex_GSReset = "41 10 42 12 40 00 7F 00 41 F7"
               sysex_GSExit = "41 10 42 12 40 00 7F 7F 42 F7"
               sysex_XGReset = "43 10 4C 00 00 7E 00 F7"
         ev.sysexdatalength   event EVENTTYPE_SYSEX contains sysex data length in
bytes
```

Hint: constants listed above should be used e.g. if (ev.type == EVENTTYPE_CONTROL && ev.controlnr == ctrl_volume) ev.controlvalue = 127;
Important: many of those event parameters can only be accessed when the current MIDI command has a compatible event type e.g. ev.velocity is only available for notes ev.type == EVENTTYPE_NOTE and else the script aborts with an error. Therefore restrict in the condition the command type e.g. ev.type == EVENTTYPE_NOTE && ev.velocity >= 80

assignment operators

```
      =    assignment of a value e.g. ev.controlvalue = 127
      +=   increment by value   e.g. if (ev.controlvalue + 10 <= 127)
ev.controlvalue += 10;
      +=   appending a string   e.g. ev.text += "\n";
      -=   decrement by value
      *=   multiplication by value e.g. if (ev.type == EVENTTYPE_TEMPO) ev.bpm *= 2;
      /=   division by value
```

Functions (more can be found in gnscript package)

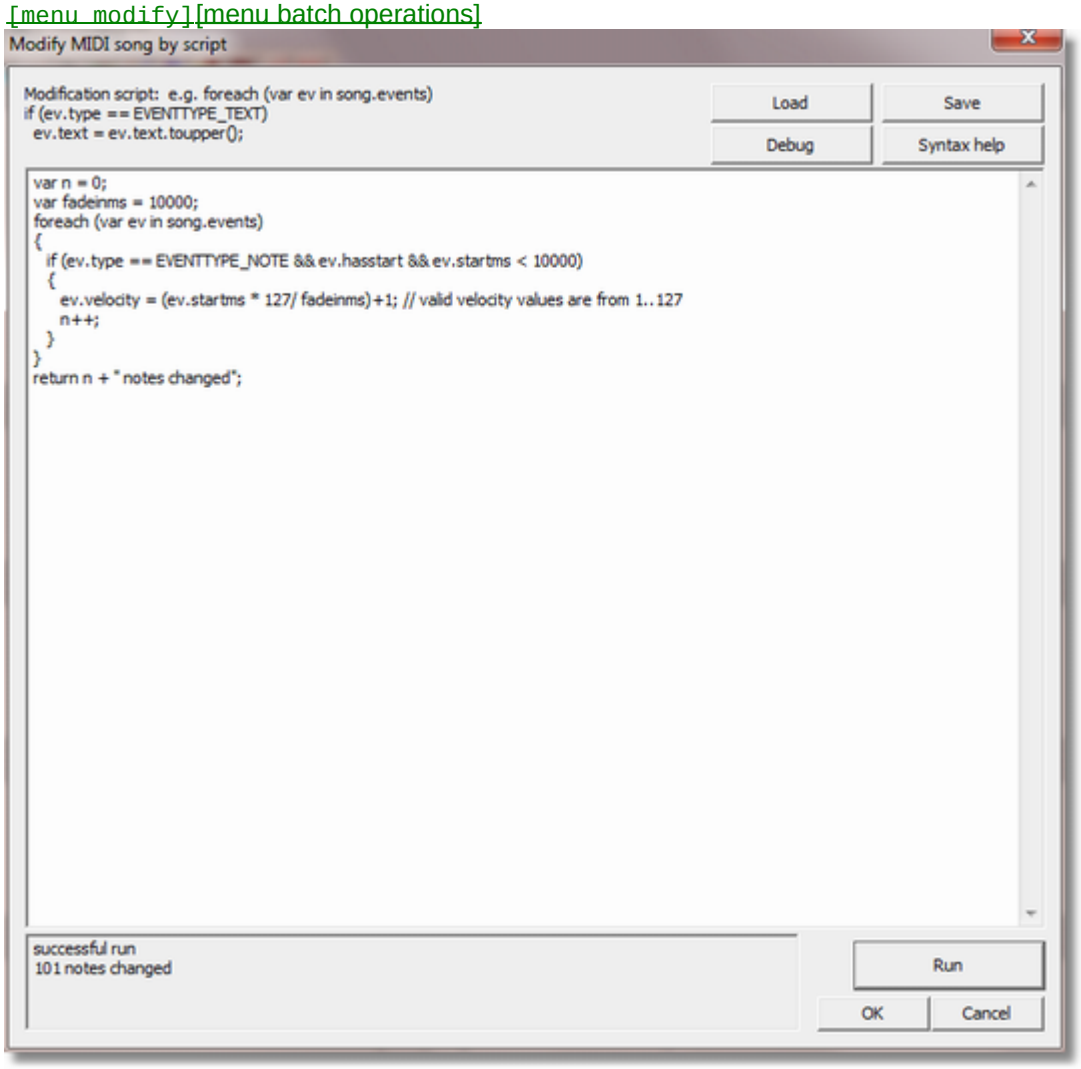
```
random(maxvalue)      a random value between 0 and maxvalue-1 e.g. ev.controlvalue
+= random(5);
replace(text, pattern, against)         replace all occurrences of pattern inside
text by against e.g. text.replace('(c)', 'copyright');
text.index(subtext)   first position of subtext inside text or -1 if not found
text.toupper()        change all characters a-z to A-Z
text.tolower()        change all characters A-Z to a-z
text.mid(pos, len)    get part of text beginning at index pos and maximum length len
text.left(len)        get part of text from beginning with maximum length len
match(text,regex, options)     match text against regular expression
substitute(text, regex, against,options)   replace parts of text matching a regular
expression regex by against e.g. text.substitute('hel.*o', 'HELLO', 'gi');
round(value)
song.deleteevent(ev.eventindex())    deletes event ev
```

Modification examples:

```
if (ev.type == EVENTTYPE_TEXT) ev.text = ev.text.toupper();
if (ev.type == EVENTTYPE_SYSEX) song.deleteevent(ev.eventindex());
if (ev.type == EVENTTYPE_CONTROL && ev.controlnr == ctrl_volume) ev.controlvalue =
max(127, ev.controlvalue + random(10));
if (ev.type == EVENTTYPE_TEXT && ev.texttype == meta_trackname) ev.text =
ev.text.replace(" ", "-");
```

3.137 Modify MIDI song by script (for experts)

[menu_modify][menu batch operations]



Modify MIDI content in current MIDI file by a C programming script. This operation requires some understanding of programming scripts in any language.

modification script
This field contains a script code with multiple statements and for/foreach/while loops and conditional if statements to describe which events needs to be changed and how they are changed.
Define variables to remember information (e.g. var n = 0;)
The script may return a result that will be displayed when running successful (e.g. return modifiedcount + " events modified")

Run
Enter modification script and use button run to start the modification.

Debug
Enter modification script and use button debug to display a debugger window that lets step through the statements and stop at breakpoints and error exceptions and lets quick evaluate an expression or enter a debugger command like printvariables or printcallstack.

Syntax errors

errors are displayed in the box below the modification edit box (in English language).

Load
Open a *.dlg text file to load previously stored settings for the dialog

Save
Saves current dialog settings into a .dlg text file for future reuse

Results:
If running script is successful then a successful information and the value returned by the script will be displayed. For syntax errors or error exceptions the error message (sometimes with line, column numbers of the source code position)

OK
closes the dialog and generates a MIDI file opened in GNMIDI which contains the modifications done by the gnscript expressions.

Syntax help
will show this page (also F1 key). Use also gnscript package at https://www.gnmidi.com to learn from script examples.
The gnscript syntax and general available script operators and functions can be found in a sub page of this article.
The MIDI functions and data structure of loaded song can be found in sub page of this article.
Loading of the MIDI file is already done by this operation and saving result will be done using OK button if script runs successful.

Example script:
```
var n = 0;
var fadeinms = 10000;
foreach (var ev in song.events)
{
   if (ev.type == EVENTTYPE_NOTE && ev.hasstart && ev.startms < 10000)
   {
      ev.velocity = (ev.startms * 127/ fadeinms)+1; // valid velocity values are
from 1..127
      n++;
   }
}
return n;
```

Too difficult
If programming is too difficult for you maybe we could program user tools for your special purpose that can be used directly in GNMIDI.
It can analyse or modify or generate MIDI files.
In fact in most cases we use the gnscript language for implementation.

3.137.1 gnscript midi modification syntax

gnmidiscript script language is an extension of gnscript with functions, constants, object types to analyse, modify, create MIDI files

This file describes the syntax elements and functions that are added for handling MIDI file access only. The basic script language elements are described in gnscript project.

CONSTANTS

defines following constants

EVENTTYPE_NOTE
EVENTTYPE_PROGRAM
EVENTTYPE_CONTROL
EVENTTYPE_HEADER
EVENTTYPE_TEMPO
EVENTTYPE_TACT
EVENTTYPE_TEXT
EVENTTYPE_PITCHBEND
EVENTTYPE_AFTERTOUCH
EVENTTYPE_POLYAFTERTOUCH
EVENTTYPE_KEY
EVENTTYPE_SYSEX
EVENTTYPE_META
EVENTTYPE_REALTIME
EVENTTYPE_ENDTRACK
version_multichannel = 0
version_singlechannel = 1
version_multitrack = 1
version_multisong = 2
COMBINE_NOTES = true
SPLIT_NOTES = false
drumchannel = 10
middlec = 60
meta_seqnumber
meta_text
meta_copyright
meta_trackname
meta_instrument
meta_lyric
meta_marker
meta_cuepoint
meta_programname
meta_devicename
meta_prefixchannel
meta_prefixport
meta_endtrack
meta_tempo
meta_smpte
meta_meter
meta_key
ctrl_highbank
ctrl_wheel
ctrl_breath
ctrl_foot
ctrl_portamentotime
ctrl_data
ctrl_volume
ctrl_balance
ctrl_expression
ctrl_effect1
ctrl_effect2
ctrl_slider1
ctrl_slider2
ctrl_slider3
ctrl_slider4
ctrl_lowbank
ctrl_lowdata
ctrl_hold

ctrl_portamento
ctrl_sustenuto
ctrl_soft
ctrl_legato
ctrl_hold2
ctrl_sound_variation
ctrl_resonance
ctrl_sound_release_time
ctrl_sound_attack_time
ctrl_xg_brightness
ctrl_xg_portamento
ctrl_reverb
ctrl_tremolo
ctrl_chorus
ctrl_xg_effect4
ctrl_phaser_level
ctrl_datainc
ctrl_datadec
ctrl_lownrpn
ctrl_highnrpn
ctrl_lowrpn
ctrl_highrpn
ctrl_allsoundoff
ctrl_allcontroloff
ctrl_localkeyboard
ctrl_allnotesoff
ctrl_omnioff
ctrl_omnion
ctrl_mono
ctrl_poly
pitch_center = 0
pitch_maxdown = -0x2000
pitch_maxup = 0x1fff
event_songpos
event_songselect
event_tunerequest
event_clock
event_start
event_continue
event_stop
event_activesense
sysex_GMReset = "7E 7F 09 01 F7"
sysex_GMExit = "7E 7F 09 02 F7"
sysex_GM2Reset = "7E 7F 09 03 F7"
sysex_GSReset = "41 10 42 12 40 00 7F 00 41 F7"
sysex_GSExit = "41 10 42 12 40 00 7F 7F 42 F7"
sysex_XGReset = "43 10 4C 00 00 7E 00 F7"
MIDI_COMPRESSION_NONE
MIDI_COMPRESSION_MEDIUM
MIDI_COMPRESSION_FULL
CHORDFORMAT_LYRIC = "LYRIC"
CHORDFORMAT_TEXT = "TEXT"
CHORDFORMAT_MARKER = "MARKER"
CHORDFORMAT_PSR = "PSR"
CHORDFORMAT_WORDBOX = "WORDBOX"
CHORDFORMAT_BIAB = "BIAB"

NEW OBJECT TYPES

MidiSong is a structure that keeps optional file name of a song and loaded song info

the structure contains only the fields (no remove, no insert):
filename a string
events a vector of MidiEvent objects

Hint: modifying tempo or tact inside for or foreach events loop is critical.
it causes that the times of all events are calculated again and resorts the order of events by time.
Use setautoupdatetimes() function to turn off/on this behaviour while working in the loop.

MidiEventList is a vector of MidiEvent objects.

It allows to insert, remove and modify elements.

MidiEvent

The MidiEvent structure object allows to get or modify information from different kind of event types.
The MidiEvent object here allows to access every existing field from any of these types (no casting required but if getting a field that does not exist will result in a runtime error)

MidiEvent can have following structure keys that are strings

type (delivers null object for null Midi Event object)
program
programname
note
notename
text
key
polykey
sysex
meta

MidiEvent can have following structure keys that are numbers

track
channel
startunit
endunit
length
startms
endms
duration
hasstart
hasend
iscombined
firstunit
lastunit
firstms
lastms
unitsperbeat
version
trackcount
bpm
microsecondsperbeat
nomin
denom
texttype

pitchbend
aftertouch
polykey
polyvalue
key
metatype
realtimeevent
realtimeparam
velocity
velocityoff
controlnr
controlvalue
sysexdatalength
metadatalength

all fields (exept type) can be modified

hint: when setting startunit, endunit, starms, endms exception may occur if the new value causes end < start . Use setnotetimes or setnoteunits for setting both values at same time.

LIVE VARIABLES

live variables are global variables that are dynamic

midicompression

this live variable (long) gets or sets MIDI compression. Live variable means that when the program changes the value the variable also changes and same in other direction.

it must have the values of constants
MIDI_COMPRESSION_NONE,MIDI_COMPRESSION_MEDIUM,MIDI_COMPRESSION_FULL

FUNCTIONS

newsong(version,resolution)
midiload(filename,combineoption)
midisave(&song,filename)
deleteevent(&song,index)
  after deleting you should call song.compact() or check before accessing events (isnullevent() or event.type == "" or event.eventindex() == -1)
collectevents(&song,andfieldname,andoperator,andfieldpattern,...)
  e.g. find all notes with channel == 10 requires triple parameters like collectevents("channel","==","10","type","=",EVENTTYPE_NOTE)
findfirstevent(&song,andfieldname,andoperator,andfieldpattern,...)
  e.g. find first note in channel 10 requires triple parameters like collectevents("channel","==","10","type","=",EVENTTYPE_NOTE)
compact(&song)
eventindex(&event)
event2string(&event)
inserttempo(&song,index,startunit,microsecperbeat)
inserttact(&song,index,startunit,nomin,demon)
inserttext(&song,index,track,startunit,texttype,text)
insertprogram(&song,index,track,channel,startunit,programnumber)
insertcontrol(&song,index,track,channel,startunit,controlnumber,controlvalue)
insertnotestart(&song,index,track,channel,startunit,note,vel)
  -- deprecated: better use insertnote(&song,index, notestructure)
insertnoteend(&song,index,track,channel,endunit,note,veloff,...)
  -- deprecated: better use insertnote(&song,index, notestructure)

insertnote(&song,index,track,channel,startunit,endunit,note,vel)
-- deprecated: better use insertnote(&song,index, notestructure)
insertnote(&song,index,track,channel,startunit,endunit,note,vel,veloff)
-- deprecated: better use insertnote(&song,index, notestructure)
insertnote(&song,index, notestructure)
notestructure can be { track: tracknr, channel: channelnr, startunit: optionalstartunit, endunit: optionalendunit, note: notename, velocity : optionalvelocityon, velocityoff: optionalvelocityoff)
parsenotename(notename)
convert a note name to a note number 0-127 if note name is valid and known (else -1)
getnotename(notenumber)
convet a note number to note name
insertpitchbend(&song,index,track,channel,startunit,pitchbend)
insertaftertouch(&song,index,track,channel,startunit,aftertouch)
insertpolyaftertouch(&song,index,track,channel,startunit,polykey,polyvalue)
insertkey(&song,index,track,channel,startunit,key)
insertsysex(&song,index,track,startunit,sysex)
insertmeta(&song,index,track,startunit,metatype,meta)
insertrealtime(&song,index,track,startunit,realtimeevent,realtimeparam)
insertendtrack(&song,index,track,startunit)
insertmeasure(&song,barnr,newtactnom, newtactdemom)
insert a new empty measure with given tact e.g. 3/4
insertmeasure(&song,barnr,newtactnom, newtactdemom,barcount)
insert a some empty measures with given tact e.g. 3/4
moveevent(&event,newindex)
copyevent(&event,newindex)
copyevent(&event,&newsong,newindex)
duplicatechannel(&song,channel,newchannel,delayunits)
duplicate all events of a channel to a new channel with some units position difference
duplicatechannel(inputfilename,outputfilename,channel,newchannel,delayunits)
setnotetimes(&event,startms,endms)
setnoteunits(&event,startunit,endunit)
timeat(&song,unit)
unitat(&song,ms)
tempoat(&song,unit)
bpmat(&song,unit)
bpm2microsecperbeat(bpm)
microsecperbeat2bpm(microsecperbeat)
setautoupdatetimes(&song,autoupdatetimes)
getautoupdatetimes(&song)
updateeventtimes(&song)
updatetimetable(&song)
tactat(&song,unit)
tactlength(&song,nom,denom)
measurelength(&song,measurenr)
measureatunit(&song,unit)
posatunit(&song,unit)
unitatmeasure(&song,measurenr)
unitatmeasure(&song,measurenr,beatnr)
unitatmeasure(&song,measurenr,beatnr,tick)
midiposition(&song,positiontext)
supports time and unit and bar position e.g. 0:30 is 30 seconds after beginning of song,
error if invalid syntax,
result is midi unit for given song
barunits(&song)
lastunit(&song)
gettrackcount(&song)
getfirsttrackforchannel(&song, channel)
find a track number that already uses channel

getresolution(&song)
getversion(&song)
loadbestchords(&song)
loadchords(&song)
loadchords(&song,format)
removechords(&song)
removechords(&song,format)
ischordevent(event,onlysafetextchords)
  identify chord event, for text onlysafetextchords decides if probably lyrics are probably not chords e.g. Am
isnullevent(event)
  a song might contain null events where events have already been deleted
removefade(&song)
containsfadein(&song)
containsfadein(&song,&fadeinfo)
containsfadeout(&song)
containsfadeout(&song,&fadeinfo)
miditool(commandline)
  e.g. midicat, midi1to0, miditrim, midi2txt, txt2midi, ...
initialisecontrol(&song,channel,controlnr)
  inserts a control of given controlnr with default control value if there is no such control before first note in channel
initialiseprogram(&song,channel)
  inserts a program change with default program value (0, grand piano) if there is no program change before first note in channel
initialisechannelsettings(&song, channel)
  inserts channel setting commands with default values if such commands are missing before first note in channel

guesskey(midifilename)
    returns number of # signs or b signs   (signs >= 0 means # signs < 0 means b)
songkeyname(signs,minor)
    e.g. signs == 0 returns song key "C"
keysigns(key)
     returns number of # signs or b signs (information stored in song)   (signs >= 0 means # signs < 0 means b)
keysigns(key,&minor)
    returns signs and sets boolean variable minor   (signs >= 0 means # signs < 0 means b)
couldbechordname(text)
    e.g. Am, C, G7, Bb, D#,...
issafetextchordname(text)
    e.g. "Am " is no safe chord
istextchord(text)
    e.g. [Cm7]
ischordevent(event)
quantise(&song,posbase,lenbase)
    e.g. quantise to base 1/16, all channels
quantise(&song,posbase,lenbase,channels)
    e.g. quantise to base 1/16, channel list as string e.g. 1-9, 11-16
quantisebase(&song,notelen)
    e.g. notelen 1/16
unescapetext(escapedtext)
    when using commandline with macro %escapetext(text) then unescapetext(escapedtext) can be used for easier getting back original text
deletepauses(&song,fromunit,units)
deletepauses(&song,fromunit,units,deleteemptynotes)
deletepauses(&song,vectorunitranges)
    unitrange is structure fromunit, units

deletepauses(&song,vectorunitranges,deleteemptynotes)
comparemidi(&midisong1, &midisong2)
comparemididetail(&midisong1, &midisong2)
comparemididetail(&midisong1, &midisong2, filename1)
comparemididetail(&midisong1, &midisong2, filename1, filename2)
equalevents(&event1, &event2)
equalevents(&event1, &event2,ignoretrackno)
equalevents(&event1, &event2,ignoretrackno,ignoretiming)
getmetatypename(metatypenumber)
  returns the name of the metatype number e.g. 3 => "Trackname"
getmetatypenumber(metatypename)
  returns the number of the meta type name e.g. "trackname" => 3
ischannelpolyphone(&song, channel, minnotesoverlapping)
  finds a note that plays at same time with other notes
ischannelpolyphone(&song, channel)
  finds a note that plays at same time with an other note (minnotesoverlapping=2)
__GNMIDISCRIPTVERSION__()
 return version string e.g. 1.0
__USEGNMIDISCRIPTVERSION__(version)
 error if version e.g. 1.0 is older than CURRENTGNMIDISCRIPTVERSION
setStatusbarText(shortmsg)
 only usable when application has statusbar control
dumpevents(&song)
 get all song data as text string
dumpevents(&song,firstindex,lastindex)
 get some events of song between index firstindex and lastindex (inclusive) as text string
replaceprogrambank(&song, channels, oldprog, oldbankmsb, oldbanklsb, newprog, newbankmsb, newbanklsb)
 find old program/bank in given channels of a song and replace it with a new program/bank
getprogrambankranges(&song,channel,track)
 find areas with sound addresses in song at channel and track
getprogrambankranges(&song,channel)
 find areas with sound addresses in song at channel
setprogrambank(&song, channel, track, startunit, endunit, prog, bankmsb, banklsb)
 set a program/bank address in channel and track from position startunit till endunit
findmatchingsoundaddress(&song, channel, addresspattern)
 find soundaddress in a channel of a song and return a vector of event indices (indexlist) for the program and bank events used,
 addresspattern is a structure { program: 3, msbbank : 0, lsbbank: -1} where bank numbers -1 are matching any bank address
replacesoundaddress(&song, channel, addressreplace, &indexlist)
 replace the found sound addresses in indexlist by a new sound address

3.137.2 gnscript Sprachelemente

gnscript v1.4 by G. Nagler 2021 (c) GN Midi Solutions

a script language

The script text is parsed into a recursive structure and interpretated without building an assembly code.
The script text may be encrypted with gnscriptcrypt.exe tool so that the file is binary and not readable for average user.
The script source may be binded with a script sum so that executing a modified script is not allowed when script sum is missing or does not match expected value.

==========
LEGAL
==========

current state of gnscript program is for demonstration and testing only.
It may not be used commercially.
No guarantees can be given at moment.
gnscript is currently tested with Windows 7, 8, 10, 11

==========
COMMENTS
==========
single line comments begin with // ... till end of line
multi line comments begin with /* and end with next */
nested comments are not allowed: e.g. START/* // /* */END ... */

==========
TOKENS
==========
LITERAL "...." or '....' or __multiline__("...") or __multiline__('...') or __noescape__('\[abc\]') or __escape__(...) or __singleline__(...);
normal literals must not contain unencoded line breaks. multiline literals can be longer and can be more lines.
backslashes and other special characters must be escaped in a literal e.g. \\ \n \r \t \x7f
using __noescape__ the literal does not escape any characters and the used quote character may not be used in the literal
__escape__(xxx) and __singleline__(xxx) are default if not specifying __multiline__ or __noescape__
__multiline__ and __noescape__ may be used together __multiline__(__noescape__('text over many lines'))
characters are same as string literals containing only one character

IDENTIFIER alphabetic character sequence that may contain underscore _ and digits inside the identifier (can not begin with digit).
identifiers are case sensitive.

NAMESPACEIDENTIFIER
an identifier for variable or function name that may optionally start with :: or could contain namespace references e.g. ::topnamespace::subnamespace::funcname(params)
namespace can be used to make a global name more specific for a certain package to avoid conflicts with system function names or global variable names.
using :::: between two names is invalid namespace identifier ending with :: is invalid namespace
(no "use namespace" statement is available, so if using name space then the full name space must always be used)

NUMBER signed long integer number -137
hexadecimal long integer number 0x3af (digits 0-9, a-f, A-F)
number with optional exponent 10e3, 10.3e-3

BOOLEAN true or false
value not 0 or 0
'yes' or 'no'
'ja' or 'nein'

CONSTANTS
null (null object)
true
false
pi
e

user applications may define more constants

```
================================================================
VARIABLES
================================================================
var anydata, value = -3, string='abc',  vector = [1,2,3], structure = { a: 13, b: 'abc' };

1. local variables
2. function argument variables
3. global variables
4. system variables

Before assigning any value to a variable it is null and has undefined type.
Use keyword auto to make sure that the variable is defined and an already existing variable is reused
(auto var maybealreadyexistingvariable;).
It gets a value and a type during assigning values.
Using a variable is in most cases a runtime error.
Some exceptions where using undefined variable values is allowed:
   operator + allows concatenating undefined values with string values (but not for adding numbers).
   ++ and -- operators assume value 0 for undefined variable value.
   == and != for comparing (possibly not initialized) variables against null
   isdefined(obj), isvariabledefined(obj) might be called using uninitialized or null variables

Variables with same name can exist in different scopes. Variables in inner scopes hide variables of
same name in outer scopes.

function func(v)
{
  // here it is v the function argument
  {
    var v = v; // local variable copies the function argument value
    // variable v is not the function argument
  }
  // here it is v the function argument again
}

================================================================
CONSTANT VALUES FOR INITIALIZING
================================================================
var stringv = 'a b c';
    functions __multiline__() and __escape__()  etc. can be used to use easier syntax
    for entering multi line text or text with or without escape sequences like \n

var number = 13;
var hexnumber = 0x3a;
var floatingnumber = 3.14; // 3. or .01 are not floating point numbers

Hexadecimal notation in literals are never converted to numbers (they remain strings).
var hexstring = '0x3a';

But you can force the conversion to a number by cast operator (int)hexstring or (long)hexstring.

var vectorv = [];
var vectorv = [1,'a',3..5]; // produces a vector that contains 1,'a',3,4,5

var structurev = {a: 13, b: 'xy'; c: '00.00'; 12: 'z'; ' ': 'blank'; }

Structure keys are strings. when using search functions for structures comparing is string equality.
```

```
================================================================
TYPES
================================================================
null   (undefined or uninitialized)
string (string or any number)
vector of objects (index values 0,1,2...size-1)
structure of objects to objects (map)
reference to an object  (usually used for reference parameters in functions)
user objects that are based on above types, implementation can define self if value or reference should
be copied during assignment or using as parameter

A variable automatically gets the type of an assigned value.
A variable type can be determined using cast type operators:
      (string)value, (long)value, (double)value, (vector)value, (structure)value

A string type expression can be also a number (long or double). It remains an unmodified string till a
number operation is used:
   var x = '0001.0';  // remains a string as long as it is not used by a number calculation
   var y = x+1;   // the x is converted to number 1 and adding 1 results in 2 (after this still can be used
as string or as number)

   var x = '0001.0';
   var y = (string)x+1;  // x is casted to string and a 0 character is added by string concatenation. Result
y is '0001.01' but can be used as string or number  same as x was used.

================================================================
EXPRESSION RESULTS
================================================================
exception    (error structure with message and source location where it occured: e.what, e.pos, e.linenr,
e.columnnr, e.filename, e.info, e.isfatalerror)
null        (no result)
boolean     true (also number != 0, 'true', 'yes', 'ja') or false (number 0, 'false', 'no', 'nein')
object      (can be casted to string)

============================
FUNCTION DEFINITION
============================
function functionname(param1,&param2, ...)

parameters are copied by default when calling a function. parameters with & are used by reference
(not copied, any changes are also in the original scope variable).

Variable arguments are proviced by l-value (not copied). The optional arguments before the variable
arguments must exist and are not necessarily l-values.

some functions have variable number of arguments (...), e.g. print('found ', count, ' items.');

When using variable arguments (...) parameter then a vector argument with name variableargs exists
and is a vector of variable parameters.
Function getfunctionargs() can be used to get a structure of all current function argument variables
including variableargs if the function has ... arguments

function myvarargsfunc(a, ...)
{
  var result = a;

  for (var i = 0; i < variableargs.size(); i++)
  {
```

```
var parami = variableargs[i];
      result = max(result, a);
  }

  return result;
}

function myvarargsfunc(a, ...)
{
  var args = getfunctionargs();
  println("all function arguments: ", args);
  println("a=", args.a);
  println("variableargs=", args.variableargs);

}
============================
CAST TYPES
============================
Following cast types can be used:
          long, double, float, int, string, structure, vector
var x = (long)'00003.14';    // x becomes a long 3

Function gettype(obj)  returns the current type of an object or expression
((long)'00003.14').gettype() returns string long

(vector)null casts to empty vector
(structure)null casts to empty structure

Casting a structure that contains key numbers 0..n-1 to a vector results in a vector of the structure
values indexed by 0..n-1 only if all index values are in the structure.
  e.g. {0:10,2:11,1:13}  casts to vector [10,13,11]

Casting an associative structure to vector results in a vector of structure elements
  e.g. {a:10,b:0,c:7} casts to vector [{a:10},{b:0},{c:7}]

============================
FUNCTION CALLING
============================
Functions can be called following 2 ways:
    funcname(param1, param2, param3);
      or
    param1.funcname(param1,param2);

Both have same meaning.
e.g. 3.14.sqr() == sqr(3.14)

Second calling method should be used when left side is a variable.
e.g. list.push(value);

Available functions are only searched by name and number of the caller arguments. It does not
consider their types or contents. It is not allowed to define a function with same name and same
number of arguments twice.
Functions can only be defined in global scope. Local functions are not supported. Function prototypes
are not supported. Available functions are known directly after parsing before executing first statement.

Functions are found in following order:
1. function definitions by function funcname(arguments)
2. user functions defined in user extensions in order of added extensions
```

3. system functions

It is allowed to redefine a function that already exists as system function e.g. function min(a,b,...) { ... return myminimum; }

==========
VECTORS
==========
Vectors are arrays of a certain size indexed by a number between 0 and vector.size()-1

```
v.size()
v[3] = value;
v.push(value); // add value as last element
value = v.pop(); // remove last element
value = v.shift(); // remove first element
v.unshift(value);  // insert value as first element
v.insert(index, value); // insert value at position index, if index == v.size() appends the value to end
```

v['text'] = value produces an entry in a structure same as v{'text} = value;
Hint: vector init supports constant numbers and number range and strings and character ranges
  e.g. var v = [3,4..7,'word','A'-'Z'];

BNF syntax
==========

```
<gnscript> ::= <statements>

<statements> ::= <statement> [ ';' <statements> ]

<statement> ::= <ifstatement>
              | <switchstatement>
              | <forstatement>
              | <whilestatement>
              | <dowhilestatement>
              | <repeatuntilstatement>
              | <foreachstatement>
              | <loopbreakstatement>
              | <loopcontinuestatement>
              | <gotostatement>
              | <gotolabelstatement>
              | <vardefinestatement>
              | <undefinestatement>
              | <functionreturnstatement>
              | <functiondefinitionstatement>
              | '{' <statements> '}'
              | <trycatchstatement>
              | <expression>

<ifstatement> ::= 'if' '(' <expression> ')' <statement>
                [ 'else' <statement> ]

<switchstatement> ::= 'switch' '(' <expression> ')'
                      '{'
                      <caseliststatement>*
                      '}'

<caseliststatement> ::= 'case' <expression> [ '..' <expression> ] ':'
                        | 'default' ':'
```

| <statement>

Hint: case 13..224: is a number range. case 'a'-'z': is a character range

<forstatement> ::= 'for' '(' <expression> ';' <expression> ';' <expression> ')'
<statement>

<whilestatement> ::= 'while' '(' <expression> ')' <statement>

<dowhilestatement> ::= 'do' '{' <statements> '}' 'while' '(' <expression> ')'

Hint: repeat statements while expression is true (or break or last or return or exception)

<repeatuntilstatement> ::= 'repeat' '{' <statements> '}' 'until' '(' <expression> ')'

Hint: repeat statements until expression is true (or break or return or exception)

<foreachstatement> ::= 'foreach' '(' ['var'] IDENTIFIER 'in' <expression> ')' <statement>

Hint: it is not allowed to assign values to the foreach variable.

<loopbreakstatement> ::= 'break' | 'last'
Hint: last and break have same meaning and exits current loop

<loopcontinuestatement> ::= 'continue' | 'next'
Hint: continue and next have same meaning to continue with the next loop step
Hint: for(init;condition;step) continues with step command
Hint: while (condition) continues with condition
Hint: foreach (var v in collection) continues with next element v from collection)
Hint: do { statements } while (condition) continues with statements (not with condition)
Hint: repeat { statements } until (condition) continues with statements

<gotostatement> ::= 'goto' IDENTIFIER

<gotolabelstatement> ::= IDENTIFIER ':'

<vardefinestatement> ::= ['auto'] 'var' NAMESPACEIDENTIFIER ['=' <expression>]

<undefinestatement> ::= ['auto'] 'undef' NAMESPACEIDENTIFIER ['=' <expression>]

<functionreturnstatement> ::= 'return' [<expression>]

<functiondefinitionstatement> ::= 'function' NAMESPACEIDENTIFIER '(' [<functionparamlist>] ')' '{'
<statements> '}'

<functionparamlist> ::= <functionparam> [',' <functionparamlist>]

<functionparam> ::= ['&' IDENTIFIER]

<trycatchstatement> ::= 'try' '{' <statements> '}' 'catch' '(' ['var'] IDENTIFIER ')' '{' <statements> '}'

<expression> ::= <assignmentexpression> [',' <expression>]

<assignmentexpression> ::= <questionexpression> [<assignmentoperator> <assignmentexpression>]

<assignmentoperator> ::= '=&' | '=' | '-=' | '+=' | '*=' | '/=' | '%=' | '&=' | '|=' | '^=' | '<<=' | '>>='

Hint: =& is reference assignment

Hint: %= is modulo
Hint: &= is bit-and assignment
Hint: |= is bit-or assignment
Hint: ^= is bit-xor assignment
Hint: <<= is bit shift left assignment
Hint: >>= is bit shift right assignment

<questionexpression> ::= <logicalorexpression> ['?' <questionexpression> ':' <questionexpression>]

<logicalorexpression> ::= <logicalandexpression> ['||' <logicalandexpression>]

<logicalandexpression> ::= <bitorexpression> ['&' <bitorexpression>]

<bitorexpression> ::= <bitxorexpression> ['|' <bitxorexpression>]

<bitxorexpression> ::= <bitandexpression> ['^' <bitandexpression>]

<bitandexpression> ::= <equalityexpression> ['&' <equalityexpression]

<equalityexpression> ::= <relationexpression> [<equalityoperator> <relationexpression>]

<equalityoperator> ::= '==' | '!=' | '=~' | '!~' | 'in'

<relationexpression> ::= <shiftexpression> [<relationoperator> <shiftexpression>]

<relationoperator> ::= '<=' | '<' | '>' | '>='

<shiftexpression> ::= <additionexpression> [<shiftoperator> <additionexpression>]

<shiftoperation ::= '<<' | '>>'

<additionexpression> ::= <multiplicationexpression> [<additionoperator> <multiplicationexpression>]

<additionoperator> ::= '+' | '-'

Hint: binary + operator prefers adding numbers than concatenating strings

<multiplicationexpression> ::= <unaryexpression> [<multiplicationoperator> <unaryexpression>]

<multiplicationoperator> ::= '*' | '/' | '%'

<unaryexpression> ::= '+' <unaryexpression>
| '-' <unaryexpression>
| '!' <unaryexpression>
| '~' <unaryexpression>
| '++' <unaryexpression>
| '--' <unaryexpression>
| '(' <castoperator> ')' <unaryexpression>
| '(' <expression> ')'
| <factor> [<postfixexpression>]

Hint: only ++ and -- and string + allow using undefined values.

<castoperator> ::= 'long' | 'double' | 'float' | 'int' | 'string' | 'structure' | 'vector'

<factor> ::= NUMBER
| LITERAL
| CONSTANT

| <functioncallstatement>
| NAMESPACEIDENTIFIER
| <structureinitexprssion>
| <vectorinitexpression>

functioncallstatement ::= NAMESPACEIDENTIFIER '(' functionparamlist ')'

functionparamlist ::= <assignmentexpression> [',' functionparamlist]

<structureinitexprssion> ::= '{' <structureinitlist> '}'

<structureinitlist> ::= IDENTIFIER ':' <questionexpression> [<structureinitlistseparator> structureinitlist]

<structureinitlistseparator> ::= ',' | ';'

<vectorinitexpression> ::= '[' <vectorinitlist> ']'

<vectorinitlist> ::= <questionexpression> ['..' <questionexpression>] [',' <vectorinitlist>]

<postfixexpression> ::=
| <structureaccessexpression>
| <vectoraccessexpression>
| '++'
| '--'

<vectoraccessexpression> ::= '[' <expression> ']'

<structureaccessexpression> ::= '.' IDENTIFIER
| '{' <expression> '}'
| '.' <functioncallstatement>

SYSTEM CONSTANTS

true=1, false=0, null=undefined
pi=3.14159265358979323846,e=2.71828182845904523536
FILE_ATTRIBUTE_READONLY,
FILE_ATTRIBUTE_HIDDEN,
FILE_ATTRIBUTE_SYSTEM,
FILE_ATTRIBUTE_DIRECTORY,
FILE_ATTRIBUTE_ARCHIVE,
FILE_ATTRIBUTE_DEVICE,
FILE_ATTRIBUTE_NORMAL,
FILE_ATTRIBUTE_TEMPORARY,
FILE_ATTRIBUTE_SPARSE_FILE,
FILE_ATTRIBUTE_COMPRESSED,
FILE_ATTRIBUTE_OFFLINE,
FILE_ATTRIBUTE_REPARSE_POINT,
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED,
FILE_ATTRIBUTE_ENCRYPTED,
FILE_ATTRIBUTE_VIRTUAL,

REFERENCES

The default behaviour of assignment is copying the content for standard data types (string, numbers, vector, structure).
The special reference assignment operator (=&) can be used to use a reference to the original object (var b =& a).

User data types defined in libraries can decide that an object is to copy by reference. It can also define other behaviour for r-value copying or forbid it. This is usually used when an object is bound to internal data in an application. The structure or vector object only gives possibilities to access or modify this internal data in a common practice instead of defining functions. E.g. openfile function fileopen(...) creates a structure that can only be used as long it remains an open file structure. Therefore copying is done by reference only. var file1 = file2;

E.g. a user application adds objects that can read and write to an own data structure or database.
It should be avoided to copy these objects since the copied structure could be big and has not anymore the same possibilites as the user application data structure (e.g. certain functions can only be called with the original objects and not with standard vector or structure).

To force to copy such an object to a standard type a compatible cast operator must be used e.g. var struct = (structure)userobj .
In most cases this is not useful.

FUNCTION DEFINITION

function functionname(param1,&referenceparam2)

functions may not be defined inside functions

FUNCTION CALLING

var ret = functionname(value1, variable2);

or identical if value1 is an object

var ret = value1.functionname(variable2);

Following order is used for calling:
1. user function if defined and matching parameter number
2. user application function
3. system function

system functions and user application functions support also variable arguments e.g. max(1,3,5,7,pi)

By default all parameters are copied to new parameter variables. Use &param in the function header to use references to the original objects.
Reference parameters modify values in the caller variables and speeds up calling by avoiding copying.

==========
system functions
==========

CODE FUNCTIONS

__VERSION__()
  gnscript version
__USEVERSION__(version)
  require a minimum gnscript version
abort(message)
abort()
die(message)
die()
  abort script optional with error message (default: empty message)
exit(exitcode)
exit()

abort script with application exit code (default exitcode 0)
throwexception(reason)
sleep(milliseconds)
  only waits
wait(milliseconds)
  waits while system can do some work
allowmessageprocessing()
  shortly wait that system can do some work
__LINE__()
  current line number
__LINE__(sublevel)
  current line number of some caller
__POS__()
  current source position
__POS__(sublevel)
  current source position of some caller
__FILE__()
  current source file name
__FILE__(sublevel)
  current source file name of some caller
__FOLDER__()
  current source file folder path
__FOLDER__(sublevel)
  current source file folder path of some caller
__FILE__(sublevel)
  current source file folder path of some caller
__CODE__()
  current source code line
__CODE__(sublevel)
  current source code line of some caller
__CODEPOS__()
  current filename and line and column position and source code line
__CODEPOS__(sublevel)
  current filename and line and column position and source code line of some caller
getdebuginfo(obj)
  get some debug information about the obj
gettype(obj)
  returns s type name for the value (can be null, long, double, string, vector, structure)
getenv(varname)
  get the value of environment variable (e.g. PATH, USERNAME)
setenv(varname,value)
  set the value of environment variable
eval(code)
  evaluate gnscript source code using current context (variables, functions available can be called). Calling an invalid code inside a script causes an exception that can be catched with try {} catch(var e) {}
gnscriptexecute(scriptpath)
  execute the script in a file (similar to eval from string).
callfunction(functionname, ...)
  calls a function by name (inner scope has priority, variable arguments are function parameters, parameter count must match with existing function)

callfunction(functionheader, ...)
  calls a function by function header (e.g. 'myfunc(arg1,...)' inner scope has priority, variable arguments contain the function parameters. Functions with not matching parameter count are ignored, if more parameters are specified in call then they are ignored e.g. 'myfunc()' will ignore extra parameters added in call.

include(filename)
  include a .gnscript source code from other file

uselibrary(libraryname,...)
  load one or more dll from given path (default extension .dll is automatically added)
  libraries search order: absolute path name, library folders from setlibrarypath(), current folder, source code folders, interpreter application folder, system executable path
  currently only the main interpreter can call libraries, libraries can not use other libraries

setlibrarypath(libfolder,...)
  add folders to library path

getlibrarysummary(libraryname)
  get information about a loaded library
  uselibrary must have been called successfully
  gives an overview about available function headers, constants, library variables
  e.g. uselibrary('gnscripttestlibrary'); println(getlibrarysummary('gnscripttestlibrary'))

TEXT FUNCTIONS

tostring(obj)
  convert obj to text
hexdump(obj)
  convert obj text to hexadecimal character numbers (can help to display binary data during debugging)
translate(text, context)
  translate text using existing .translate files, use context for translating same text different for different purposes
translate(text)
  translate text using existing .translate files, use global translation context
strlen(text),
length(text),
size(text)
  length of text (multicharacter encoded, number of total bytes)
reservespace(text,len)
  when appending something often to text then reserving space might speed up the operation because reallocation memory is not necessary every time
comparetext(a,b)
   returns -1 if a < b or 0 if a == b or 1 if a > b
comparetextignorecase(a,b)
   returns -1 if a < b or 0 if a == b or 1 if a > b for uppercase text a and b
comparenumeric(a,b)
   returns -1 if a < b or 0 if a == b or 1 if a > b for numbers a and b or text a and b (if a or b is no number)
getarg(varname)
getarg(varname,defvalue)
  safe method to get variable by name, get empty string or defvalue if variable was not set
  for command line arguments the options can be used:
     -myoption1=myvalue getarg("myoption1") delivers string object "myvalue"
     -myoption2 getarg("myoption2") delivers boolean object true
     input.mid output.mid getarg("param1") delivers string input.mid and getarg("param2") delivers string output.mid
                getarg("scriptfilename") delivers the script name parameter
  without using commandline the function getarg can be used to access global variables already set by the application by before the script has started
  e.g. getarg("input") returns the value of global variable input using input in the script would cause a syntax error if variable input is not defined. Alternatively isvariabledefined("input") could be asked before using variable input.

getargs()
  returns a structure that contains all options with values and parameters param1, param2 ... and scriptfilename
getglobalvariables()
  returns a structure with global variables and references to that variable values
getfunctionargs()
  returns a structure with function arguments, variable args are in member with key variableargs
mid(text,pos)
  copy characters from position pos till end of text
mid(text,pos,len)
  copy max len characters from position pos of text
left(text,len)
  copy max first len characters from text
right(text,len)
  copy max len characters from end of text
fill(text, pos, size, filltext)
  fills all characters of text inside the given area by filltext
index(text,subtext)
  find first matching character position of subtext inside text
  -1 if subtext is not found inside text
index(text,subtext, offset)
  find first matching character position of subtext inside text beginning at offset or behind
  -1 if subtext is not found inside text
containstext(text,subtext)
textcontains(text,subtext)
  check if text contains subtext
textstartswith(text,subtext)
  check if text begins with subtext
textendswith(text,subtext)
  check if text ends with subtext
rindex(text,subtext)
  find last matching character position of subtext inside text
  -1 if subtextis not found inside text
rindex(text,subtext,offset)
  find last matching character position of subtext inside text at offset or before
  -1 if subtextis not found inside text
strcmp(text1,text2)
  compare text1 and text2 case sensitive
  0 if identical
  <0 if first different character of text1 is before character in text2
  >0 if first different characterof text1 is after character in text2
stricmp(text1,text2)
  compare text1 and text2 case insensitive (A is same as a)
  0 if identical
  <0 if first different character of text1 is before character in text2
  >0 if first different characterof text1 is after character in text2
trim(text)
  remove spaces at beginning and end of text
  space character, new line, carriage return, tab character are spaces
trimleft(text)
  remove spaces at beginning of text
trimright(text)
  remove spaces at end of text
shortcut(text, maxlen)
  create a shorter text by cutting the text and adding ...
shortmiddlecut(text, maxlen)
  create a shorter text by cutting and inserting ... in middle
escapestring(text)

escape all special characters using \ sequences (\t for tab, \n for newline, \r for return, \\ for backslash ...)
unescapestring(text)
replaces escaped sequences against special characters e.g. \t against tab char \n against newline \\ against backslash ...
quotestring(text)
add quote characters (") before and after the text if the text contains special characters or the quote character self.
unquotestring(text)
if string starts with quotes " or ' then quotes are removed and text unescaped (text might contain e.g. \" as escaped character)
ord(text)
calculates the character number of first text character
text should have length 1
isnumber(text)
check if text is a number (integer or decimal)
ishexnumber(text)
check if text is a hexadecimal number (only uses characters 0-9 and a-f,A-F)
isalpha(text)
check if all text characters are ascii characters
isalphanumeric(text)
check if all text characters are ascii letters or digits
iswhitespace(text)
check if all text characters are spaces, tabs, newlines, linefeed
chr(value)
convert the character number to a string that contains this character
toupper(text)
convert all text characters to upper case (a-z to A-Z)
tolower(text)
convert all text characters to lower case (A-Z to a-z)
split(text)
split text into vector of fields assuming separator ,
split(text,separator)
split text into vector of fields using separator
splitlines(text,trimends)
split text into vector of lines (newline remains at end of each line)
if trimends is true all lines are trimmed at end
splitwords(text)
split text into a vector of words using spaces, tabs, newlines comma semicolon colon as default word separator
splitwords(text,delimitercharacters)
split text into a vector of words using the delimiter characters in the delimitercharacters string parameter
like(text,pattern)
check if text matches the pattern
pattern can contain wildcards * (any characters) ? (any single character) # (any digit)
join(separator,vector)
join all elements in vector together to one text using separator between
join(separator,text1,text2,...)
join all specified text together to one text using separator between
concat(vector)
concatenate all vector elements together without separator
concat(text1,...)
concatenate all text together without separator
replace(text,from,against)
replace all matching from parts inside text against an other text
defaultreplaceoptions()
returns a structure with default replace settings used in replace(text,from,against)

replace(text,pattern,replace,options)
  replace all matching from parts inside text against an other text
  options is an optional structure with optional boolean elments ignorecase, regularexpressions, wordsonly, wholeonly, replaceall, encodenotascii as delivered by function defaultreplaceoptions()

parseintrangelist(text)
  reads a list of numbers or number ranges and writes it into a vector (e.g. 1,2 3;5-7,9 is vector [1,2,3,5,6,7,9]

CSV FUNCTIONS

makecsvline(stringvector,fieldseparator)
   build a csv line string using allowed separator characters fieldseparator separfrom the data fields in vector stringvector
parsecsvline(csvline, fieldseparator)
   converts a valid csvline into a vector
   fieldseparators contains the possible csv separator characters e.g. ",;"
parsecsvline(csvline)
   converts a valid csvline into a vector
   field separators can be: , ; : tab
splitcsvlines(text, quotecharacters)
   split csv text using possible quote characters in quotecharacters to csv lines
   a csv line may be over multiple lines when using quote character
   quote characters are usually " and '   quotecharacters="\""
splitcsvlines(text)
   split csv text using possible to csv lines
   a csv line may be over multiple lines when using quote character
   possible quote characters are: " and '

JSON FUNCTIONS

parsejson(text)
  converts a text in format JSON to an object e.g.
createjson(obj)
  converts a structure or vector or string to JSON text e.g. defaultreplaceoptions() are converted to JSON:

```
{
   "encodenotascii": 0,
   "ignorecase": 0,
   "regularexpressions": 0,
   "replaceall": 1,
   "wholeonly": 0,
   "wordsonly": 0
}
```

REGULAR EXPRESSION FUNCTIONS

match(text,regex,...)
  check if text matches a regular expression regex
  first optional argument contains variable for the whole match
  next optional arguments contains variable for group match enclosed in regex using (and)
match2(text,regex,options,...)
  check if text matches a regular expression regex
  options are letters in the string: i for ignore case, w for words only, g for global
  first optional argument contains variable for the whole match
  next optional arguments contains variable for group match enclosed in regex using (and)
Hint: usually the regular expression is written in a literal so the backslash characters must be escaped. Always think what should a variable after assignment contain so that this text can be used as regular

expression.
e.g.
```
   var wholematch, devicename;
   var regularexpression = "\\[device name\\]\\s*\(.*\)$"; // \[device name\]\s*(.*)$
   if (line.match(regularexpression, wholematch,devicename))
   {
     // use devicename
   }
```
   \\[and \\] are required because the characters [and] should occur and not a regular expression character range.
   \\s in literal (\s regular expression) means any space character (space or tab or newline or linefeed)
  \\(would only match the character (and not define a group

substitute(text,regex,against)
  replace longest matching regex pattern against new text
  regex may reference groups using (...)
  against may reference groups used in regex $1 $2 ... ($0 references whole match)
substitute(text,regex,against,options)
  replace first matching regex pattern against new text using option character list
  option i uses case insensitive matching (default is case sensitive)
  option g uses replacing all matching parts (default is replace first longest matching part)
  regex may reference groups using (...)
  against may reference groups used in regex $1 $2 ... ($0 references whole match)

OBJECT FUNCTIONS

isdefined(variable)
isdefined(variablename)
isvariabledefined(varname)
  checks if variable is defined
isempty(obj)
  tests if the object is empty (can be a string or vector or structure or or null)
clear(variable)
   set variable value empty
duplicate(obj)
  create a copy of an object

COLLECTION FUNCTIONS

push(vector,value)
  append a copy of a value to a vector object
pushreference(vector,refvalue)
  append a reference to a vector object
pop(vector)
   removes an item from the back side of vector
shift(vector)
   removes an item from the front side of vector
insert(vector,index,value)
   inserts value at position index of vector
unshift(vector,value)
   inserts value at front side of vector
containskey(structure,key)
   check if structure contains key
containsvalue(obj,value)
   check if vector or structure contains this value
   check if string contains value as substring
removekey(structure,key)

remove key from structure
  remove element with key index from vector
removevalue(structure,value)
  remove structure elements matching value
  remove vector elements matching value
  remove first substring matching value from string
collectionkeys(structure)
  get a vector of all key names from the keys used in structure
collectionvalues(structure)
  get a vector of all values used in the structure
grepkeys(structure,operator,value,...)
  search key names of a structure that matches specified value compares
  compare operations are combined using AND and evaluated from left to right
  compare operators are:
    equal: ==, =, (default: empty assumes equal operator)
    not equal: !=,!,<>
    relational: >,>=,<,<=
    in collection: in,!in
    regular expressions: =~,!~
grepvalues(collection,operator,value,...)
  search key values of a structure or vector that matches specified value compares
  compare operations are combined using AND and evaluated from left to right
  same operators as in grepkeys
sort(vector)
  sort all values of a vector numerically and alphabetic
sortnumeric(vector)
  sort all values of a vector numerical (if values are numbers),data that are not sortalphabetic(vector)
  sort all values of a vector alphabetic
  numerical are compared alphabetic ignoring case letters
      23a is no number,-23,-23.4 are numbers
sortignorecase(vector)
  sort all values of a vector alphabetically ignoring case letters A == a
sort(vector,comparefunctionname,...)
  sort all values of a vector using an existing function with given name that has two arguments
  empty comparefunctionname compares numerical
  compare function must compare two objects and return true if the first value should be before the second value (lessthan implementation)
  the user function should use reference arguments lessthan(&a,&b) that unnecessary copying is avoided
inversesort(vector)
  sort all values of a vector numeric and alphabetically in inverse order (Z to A)
inversesortalphabetic(vector)
  sort all values of a vector alphabetically in inverse order (Z to A)
inversesortnumeric(vector)
  sort all values of a vector numerical (if values are numbers) in inverse order (9 to 0)
inversesortignorecase(vector)
  sort all values of a vector alphabetically ignoring case letters A == a in inverse order
inversesort(vector,comparefunctionname,...)
  sort all values of a vector using an existing function with given name that has two arguments in inverse order
  empty comparefunctionname compares numerical
  compare function must compare two objects and return true if the first value should be before the second value (lessthan implementation)
  the user function should use reference arguments lessthan(&a,&b) that unnecessary copying is avoided
  extra parameters might be added to the sort function after the function name parameter
  they will be added when calling the compare function
  e.g. sort(v,mycompare,compareoptions);

```
function mycompare(&a,&b,&compareoptions) { return a < b; }

UNITTEST FUNCTIONS
unittest_clear()
  reset the previous unit test results and counter
unittest_count()
  get the number of tests called e.g. unittest_areequal(value, correctvalue, testname)
  this count can be used to guarantee that all unit tests of a script are really executed
  e.g.  last unit test can be unittest_areequal(unittest_count(), 10)  if the module should execute 10 unit
tests
unittest_results()
  get the failed unit test result (each line looks like unit test TESTNAME failed:  reason)
unittest_areequal(value, correctvalue, testname)
  test fails if string converted value does is not equal to string correctvalue
unittest_areequalvalues(value, correctvalue, testname)
  test fails if long value is not equal to long correctvalue
unittest_arenearvalues(value, correctvalue, maxdistance, testname)
unittest_areequalbool(value, correctvalue, testname)
  test fails if boolean value is not equal to boolean correctvalue
unittest_mustcontaintext(value, text, testname)
  test fails if value does not contain text
unittest_isinvaluelist(value, correctvaluevector, testname)
  test fails if value is not equal to one of the vector elements
unittest_isinvaluelist(value, correctvaluewithseparator, testname)
  list of correct values is separated by character | . Test fails if value is not equal to one of these
values.
unittest_mustcontaintext(value, text, optionignorecase, testname)
  test fails if value does not contain text considering optionignorecase true or false
unittest_istrue(value, testname)
  test fails if value is not true
unittest_isfalse(value, testname)
  test fails if value is not false
unittest_fileexists(filepath, testname)
  test fails if a file at filepath is not existing (or accessible by this path)
unittest_arefilecontentsequal(filepath1, filepath2, testname)
  test fails if one of the files do not exist or are not readable or their binary content is different.
unittest_fail(message, testname)
  test fails always with the reason specified in message

DATE/TIME FUNCTIONS
timeseconds()
  seconds since 1.1.1970
timemilliseconds()
  milliseconds since last system start
getdatestring()
  get current local date in format d.m.yyyy
gettimestring()
  get current local time in format h:mm:ss
gettimestring(seconds)
  format seconds into format h:mm:ss
getdatetime()
  get current local date and time as structure containing keys: day, month, hour, minute, second
datetoday()
  get date structure with keys day, month, year for current local date
parsedate(text)
  get date structure from date string text e.g. 27.7.2024 or 7/27/2024
dategerman(date)
  get a date string for date structure or date string in german date format e.g. 27.7.2024
```

dateenglish(date)
  get a date string for date structure or date string in english date format e.g. 7/27/2024
dateweekday(date)
  get week day index number for date structure or date string. 0 is Sunday, 6 is Saturday
isvaliddate(date)
  returns true if date structure or date string is a valid formatted and existing date e.g. 29.2.1967 is not a valid date
daysbetweendates(date1,date2)
  returns number of days between date1 and date2. 0 if date1 is equal to date2. negative if date2 is before date1
isdatebefore(date1,date2)
  returns true if date1 structure or date string is before date2 structure or string
parsetime(text)
  get time structure from time string text e.g. 13:27:05 or 00:59
time2seconds(time)
  calculates seconds since 0:00:00 from a given time structure or time text
seconds2time(seconds)
  calculates time structure from time since 0:00:00
time2shorttext(time)
  calculates mm:ss from a given time structure or time text, if hour is not 0 then result is h:mm:ss
time2text(time)
  calculates h:mm:ss from a given time structure or time text

OUTPUT FUNCTIONS
Hint: all file operations require permission of the application that supports the operation: r=allow read file, w=allow write file, i=allow get file information

sprintf(format,...)
  format a text that may contain formatting and optional parameters e.g. %03d (similar to C sprintf)
  %% is percent character
  %d integer numbers, %l long numbers, %s string, %c character, %f decimal numbers, %x hexadecimal numbers
printf(format,...)
  print formatted text to standard output (commandline only)
print(text,...)
  print text to standard output (commandline only)
println(text,...)
  print text and newline to standard output (commandline only)
println()
  print newline to standard output (commandline only)
printerrorf(format,...)
  print formatted text to standard error (commandline only)
printerror(format,...)
  print text to standard error (commandline only)
printerrorln(text,...)
  print text and newline to standard error (commandline only)
printerrorln()
  print newline to standard error (commandline only)
fileexists(path)
  check if a file at path really exists
filesize(path)
filesize(file)
  get file size of file at path or of the open file
equalfiles(path1,path2)
  check if content of two files is binary identical
closedfilereadtext(path)
  read whole text from a file at path into one string
closedfilereadlines(path)

read all text lines from a file at path into a string vector
closedfilereadbinary(path)
  read whole data from a binary file at path into a string
closedfilewritebinary(path,buffer)
  write binary data buffer into a file at path (overwrite existing file)
closedfilewritetext(path,text)
  write text into a text file at path (overwrite existing file)
closedfilewritelines(path, lines)
  write a vector of lines to a text file
closedfileappendtext(path,text)
  write text into a text file at path (append to existing file)

fileopen(filename)
  open binary file for reading only
  returns a file handle of open file or exception when file cannot be opened
  file should be closed using openfileclose(file)
fileopen(filename,mode)
  open file according to mode
  mode w is write (overwrites file if existing, creates file if not existing)
  mode a is append (creates file if not existing)
  mode r is read
  returns a file handle of open file or exception when file cannot be opened
  file should be closed using openfileclose(file)
openfileclose(file)
  close an opened file
openfilepos(file)
  get current absolute file position of an open file
openfilesize(file)
  get number of bytes in an open file
openfileseekbegin(file)
  seek to beginning of an open file
openfileseekbegin(file,pos)
  seek current position to absolute pos of an open file
openfileseekcurrent(file,pos)
  seek current position to relative pos (can be also 0 or negative) of an open file
openfileseekend(file)
  seek end position of an open file
openfileseekend(file,pos)
  seek current position to a position relative to end (can be also 0 or negative)
openfilechangesize(file,size)
  truncate or extend file size of an open file
openfilereadbinary(file,size)
 read size bytes from an open file beginning at current file position
openfilereadline(file)
 read a line that ends with newline or end of file from an open file beginning at current file position
openfilewritebinary(file,data)
 write binary data to an open file at current file position
openfilewritetext(file,text)
 write text to an open file at current file position (newlines are expanded to \r\n)
copyfile(destpath, srcpath)
  copy file at srcpath to file at destpath (overwrite existing file), returns true if successful
copybinaryfile(destpath, destwritepos, srcpath, srcpos)
  copy a part of a binary file to a destination binay file at given position
movefile(destpath, srcpath)
  move file at srcpath to file at destpath
copybinaryfile(destpath, destwritepos, srcpath, srcpos, copysize)
  reads binary part from srcpath and writes it into file destpath at position destwritepos, returns true if successful

deletefile(path)
  delete file at path if existing (use system basket)
deletefileforever(path)
  delete file at path if existing (delete forever without basket)
createfilebackup(srcpath)
  create a backup of src file in current folder with original datetime in name
createfilebackup(srcpath, destfolder)
  create a backup of src file in destfolder with original datetime in name
isbackupfilename(srcpath)
  check if file name has format used by createfilebackup

PATH FUNCTIONS

gettemporaryfolder(...)
  get system temporary folder path
  append all specified parameters using joinpath
gettemporaryfilename(nameprefix,ext)
  get a not existing temporary file name that uses name prefix and file extension e.g.
gettemporaryfilename("begin", ".txt")
  the filename will get a random numbering
gettemporaryfilename(ext) // e.g. gettemporaryfilename(".mid")
  get a not existing temporary file name that uses file extension e.g. gettemporaryfilename(".txt")
  the filename will get a random numbering
gettemporaryfoldername(nameprefix,ext)
  get a not existing temporary folder name that uses name prefix and file extension e.g.
gettemporaryfoldername("begin", ".txt")
  the filename will get a random numbering
gettemporaryfoldername(ext)
  get a not existing temporary folder name that uses file extension e.g. gettemporaryfoldername(".txt")
  the filename will get a random numbering
getpersonalfolder(...)
getmydocumentsfolder(...)
  get my documents folder (is same as personal folder)
  append all specified parameters using joinpath
getprogramfolder(...)
  get program folder
  append all specified parameters using joinpath
getapplicationdatafolder(...)
  get application data folder
  append all specified parameters using joinpath
joinpath(vector)
  append all vector elements using backslash separator
joinpath(path,...)
  append all parameters using backslash separator (e.g. joinpath(folder, "subdirectory", "filename.ext")
getfilename(path)
  get file name from path
  assumes that path is not only a folder path
getdirectory(path)
  get folder part from path without file name
getfileextension(path)
  get the last file extension from the path (e.g. .txt)
removefileextension(path)
  remove the last file extension from path (e.g. .txt)
quotepath(path)
  add quote character (") before and after the path if the path contains spaces or special characters
getfullpath(shortpath)
  get a full path for a short path or a relative path   e.g. getfullpath('..\\subdir');
readdirectory(path)

get the directory entries of a folder as vector of entries
  each entry is a full path of a file or sub folder
createdirectory(path)
  create folder at path
createdirectories(path)
  create folders that build the path
deletedirectory(path)
  delete folder at path
directoryexists(path)
  check if folder exists at path
fileattributes(path)
  get file attributes of path
fileisnewer(path1,path2)
  return true if file at path1 is newer than file at path2 (exception if files do not exist or access to file information is not allowed)
logmessage(text1,...)
  append text to global log file
  all specified arguments are concatenated
  global log file path is set by the application

PROFILE FUNCTIONS
writeprofilestring(inipath,section,key,value)
  set .ini entry key inside section to value in file inipath
getprofilestring(inipath,section,key,defaultvalue)
  get .ini value of entry key inside section in file inipath
  if value is not set then use defaultvalue
getprofilestring(inipath,section,key)
  get .ini value of entry key inside section in file inipath
  if value is not set then use empty string
writeprofilebool(inipath,section,key,value)
  set .ini entry key inside section to boolean value in file inipath
getprofilebool(inipath,section,key,defaultvalue)
  get .ini boolean value of entry key inside section in file inipath
  if value is not set then use defaultvalue
getprofilebool(inipath,section,key)
  get .ini value of entry key inside section in file inipath
  if value is not set then use false

MATHEMATICAL FUNCTIONS

odd(value)
  check if value is odd (1,3,5,7..)
round(value)
  round value to next integer number (for fractional part 0.5-0.999 round up, else round down)
round(value,digits)
  round value to next decimal number that has number of digits in fractional part
random(maxvalue)
  get a random value between 0 and maxvalue-1
random()
  get a random value between 0 and 65535
startrandom(initrandom)
  by default random() generates random numbers. Use startrandom(constantnumber) to generate (pseudo random) number list during testing (e.g. startrandom(0); random(1000000) produces 38). For unit tests random results are not so easy to verify.
abs(x)
  -x if x is a negative number
min(vector)
  minimum of all values in the vector

min(x,y,...)
  minimum value of all arguments
  if parameter is a vector then also all the vector element values are used too
  e.g. min(7, [4,5,6]) results in 4
max(vector)
  maximum of all values in the vector
max(x,y,...)
  maximum value of all arguments
  if parameter is a vector then also all the vector element values are used too
  e.g. max(3, [4,5,6]) results in 6
sum(x,y,...)
  sum value of all arguments
average(x,y,...)
  average value of all arguments
swap(&x,&y)
  swap content of two variables
acos(x)
  arcus cosinus value of x
asin(x)
  arcus sinus value of x
atan(x)
  arcus tangens value of x
atan2(x,y)
  angle between ray to (x,y) and positive x-axis
cos(x)
  cosinus of value x
cosh(x)
  hyperbolic cosinus of value x
exp(x)
  base (constant e = 2.718) exponented by value x
log(x)
  logarithm of value x using base e (2.718)
log10(x)
  logarithm of value x using base 10
sin(x)
  sinus of value x
sinh(x)
  hyperbolic sinus of value x
tan(x)
  tangens of value x
tanh(x)
  hyperbolic tangens of value x
sqrt(x)
  square root of positive value x
sqr(x)
  x * x
ceil(x)
  rounding up to next higher integer number
floor(x)
  rounding down to next lower integer number
fract(x)
  part behind dot of value x
div(x,y)
  integer division result of x / y
modf(x)
  fractional part of value x
fmod(x,y)
  floating point remainder of x / y

pow(x,y)
  x exponented by value y (pow(2,3) = 8)

BINARY BUFFER FUNCTIONS

buffergetbyte(&buffer,pos)
  get a single byte value inside buffer at byte position pos
buffersetbyte(&buffer,pos,value)
  set a single byte to value inside buffer at byte position pos
bufferfillbytes(&buffer,pos,len,value)
  set next len bytes to same value inside buffer at position pos

DIALOG FUNCTIONS

messagebox(text, caption, boxtype)
  show a message box with caption and text
  box type can be: error, warning, information, ok, okcancel, yesno, yesnocancel, question, exclamation , abortretryignore, retrycancel or a number
  message box returns answers: yes, no, ok, cancel, retry, abort, ignore or a number
messagebox(text, caption)
  show an OK message box with caption
messagebox(text)
  show an OK message box with caption Information
getinputlinebox(prompt, caption, &inputtext)
  show a message box with an edit text box and ok and cancel button. Returns 1 for ok and 0 for cancel and sets inputtext to the user input.
filechooser(initpath, caption, options, defaultext, filter)
  show a file chooser dialog for selecting a file or folder using caption and optional initial path
  options can be a text list: save, open, multi, pickfolders, overwriteprompt, pathmustexist, filemustexist
  default extension defaultext is added automatically when no file extension was specified in the dialog
  filter is a list of pairs of caption|fileextensions e.g. Images|*.gif;*.png;*.jpg|All files (*.*)|*.*||
  filter must end with ||
  filter fileextensions must be separated by ;
  returns selected path
  returns null when nothing is selected
  when using multi option a vector of path is returned

PROCESS FUNCTIONS
findapplication(appname)
  get the full path of an application if it is found in path, empty string if it is not found
shellexecutecommandline(cmdline)
shellexecutecommand(cmdline)
  start a command line e.g. shellexecutecommandline("notepad2 mytextfile.txt"), it may also contain a DOS command e.g. del /S /Q *.bak
shellexecute(prog)
shellexecute(prog, params)
  open prog with optional params in a system shell e.g. shellexecute("notepad", "file.txt");
runprocess(cmdline)
  execute a commandline process, returns exit code   (0 is successful)
runprocess(cmdline,&processoutput)
  execute a commandline process and store the stdout text from process in processoutput variable, return exit code (0 is successful)

SOUND FUNCTIONS

beep()
  simple system beep sound

beep(soundname)
  sound names are simple,ok,warning,information,error,question or a number of a system beep sound

playmidi(events,...)
  play midi events (e.g. playmidi('144 60 127');sleep(500);playmidi('144 60 0'); plays a C note for half second

saytext(text)
saytext(text,options)
  speaks text using system speech (depending on installed text to speech driver it can be language dependent or poor quality)
  options can be 'usespeechsynth' or 'usesapi' (default)

CLIPBOARD FUNCTIONS
setclipboardtext(text)
  puts the text into Windows clipboard
getclipboardtext()
  gets current Windows clipboard text (empty string if not available)

GRAPHIC FUNCTIONS (still experimental)

picturecreate(width,height)
  create an empty picture with width x height pixels
picturedestroy(picture)
  free the picture
pictureload(filename)
  load a picture from existing file (can be .bmp, .png, .gif, .jpg, .tif)
picturesave(picture, filename, format)
picturesave(picture, filename)
  save picture to a file (overwrites existing file)
  format can be "BPM", "TIFF", "JPEG", "GIF", "PNG" or empty or omitted
  if no format is given then file format is chosen from file extension (.bmp, .tif, .jpg, .gif or .png) or default format is "BPM".
 picturesavebpm(picture, filename)
  save picture to a .bpm file
picturefillrect(picture,x,y,width,height,color)
  fill a rectangle part of a picture with a solid color
picturedrawtext(picture,x,y,width,height,text,color)
  draw a colored text into a rectangular area of a picture
getcolor32(alpha,r,g,b)
  get an rgb color with alpha channel (transparency level, 0= not transparent till 255=full transparent)
getcolor24(r,g,b)
  get an rgb color that is not transparent
getred(color)
  get red part of rgb color
getgreen(color)
  get green part of rgb color
getblue(color)
  get blue part of rgb color
getalpha(color)
  get alpha part of transparent rgb color

LICENSE FUNCTIONS
getlicensestate()
  check if a valid license is installed (alternative a demo mode could run if supported by the script)
haslicensefeature(featurename)
  check if license supports an extra feature by name (e.g. use of certain operations)

GNSCRIPT DEBUGGER

gnscript contains an embedded debugger when started with -d (-debug) option

gnscript -d filename.script

By default a Windows debugger dialog appears when the running stops at a breakable line or exception or breakpoint. By default it displays the results to a debugger window. The setting useconsole can be set to true to display the results in console.

With setting usegnscriptdebugger=false the command line debugger can be used where the debugger can be directed using text input commands that must be finished with a return key.

The debugger stops at first source code line and shows the line.
It waits for user input.

help,h shows possible input commands
return,r continue till end of function
list,l show a list of source code lines at a given line number
continue,c continue till a breakpoint stops or an exception occurs
quit,q exit script
stepinto,s execute next command and if it is a function call step into the function
stepover,n execute next command and if it is a function call step over the function
breakpoint,b set a manual a conditional breakpoint at a line
variable,v set a variable change breakpoint
print,p evaluate a script line and print the result on console

GNSCRIPT PROFILER

gnscript contains an embedded profiler when started with -profile=sortmethod
The profiler counts every code line that was executed and measures the time that the execution tooks. The results are written to gnscript.log at end of script.

sortmethod are: count, sumbrutto, sumnetto, singlebrutto, singlenetto (default: sumbrutto)
sortmethod can start with + (ascending) or - (descending) to change the sort order (default: -, descending)

This can help to optimize a slow script to find where the most time of the script is lost.

Hint: in many cases slow scripts are caused by unnecessary copying of function arguments. use reference parameters (&) if possible.

Hint: if a script runs very very slow look also into process task list if unnecessary processes are running (e.g. old instances of gnscript).

GNSCRIPT LIMITATIONS

function recursive calls are limited to a number specified by the application to prevent from possible endless recursive calls (default: 100). Allowing too many recursive calls can lead to a application stack overflow.

for, while, until loops counts are limited to a number specified by the application to prevent from possible endless loop (default: 10000).

using prefix and postfix operators in same expression are not allowed (postfix operator creates a r-value result that cannot be modified)

GNSCRIPT INTERPRETER

```
gnscript script "var x = 1; ++++x;"
interpreter result:
3
```

executes the script and displays the result value (converted to string) or error exception if an error occured

GNSCRIPT SCANNER

```
   gnscript -scanner script "var x = 1; ++++x;"

scanner result:
console L1C2 VARIABLE var
console L1C6 IDENTIFIER x
console L1C8 =
console L1C10 NUMBER 1
console L1C11 ;
console L1C13 ++
console L1C15 ++
console L1C17 IDENTIFIER x
console L1C18 ;
```

displays the scanner result token list with their source code positions or a syntax error exception if an invalid token was found

GNSCRIPT PARSER

```
   gnscript -parser script "var x = 1; ++++x;"

parser result:
STATEMENTS{
  VAR  x
    CONSTVALUE 1
  ASSIGNMENT +=
    ASSIGNMENT +=
      VARIABLE x
      CONSTVALUE 1
    CONSTVALUE 1
}
```

displays the parser result recursive grammar structure or an error exception if a syntax error was found

EXTENDING GNSCRIPT LANGUAGE

An application that uses gnscript library can override class GNScriptContext to extend the language.

CONSTANTS

own constants can be defined for own purposes e.g. EVENTTYPE_TEXT to string value 'T'

FUNCTIONS

the available system functions list can be extended by own functions. The own functions also support variable parameters list and return an object value (can also be a vector or structure or null) and error message if the call fails for any reason (e.g. parameters are not valid).

e.g. setStatusbarText(shortmsg)

LIVE VARIABLES

the application may extend the language by new object type variables. Basic types can be overrided to quicklier define own structures.

By default the live variables are used by reference. A clock variable could be defined and each

access to the variable delivers current time.
Live variables help to bind gnscript objects to existing data structures in the application. The original data is not duplicated in the script language. The original data is accessed during object usage e.g. song.event[i] =

based string objects getter and setter for string value can be defined
number objects getter and setter for numeric value can be defined
structure field names can be defined and getter and setter for each member can be defined, remove and insert key can be defined
vector getter and setter for each indexed key can be defined, remove and insert index can be defined

LIBRARIES

a DLL can implement a library extension.

The main application (interpreter) converts calls to a library extensions to calls to dll functions. Objects are converted to global data and on dll side automatically converted back. The DLL can create own object types by implementing base classes.
Such objects use the derived flag. These results objects are not copied. The interpreter remembers the class instance on the DLL side and does not hold any data copies. Everytime the DLL library must be asked and the DLL implementation can react on changes (e.g. a vector object in the library object is not visible to the interpreter but gives functions to access the vector e.g. ask vector size, ask existing vector element, pop, push, and when the interpreter does not use the object anymore it automatically calls a dll function that destroys the created object instance.

GNSCRIPT UNIT TEST EXAMPLE

==== content of file goto.gnscript
```
================================================================================
function testjumpforward()
{
  var x='test';
  goto forwardsamelevel;
  x = 'wrong';
  return x;
  forwardsamelevel:
    x = 'correct';
    return x;
}

function testjumpbackward()
{
  var x='test';
  goto forwardsamelevel;
  return x;

  backwardsamelevel:
    x = 'correct';
    return x;

  forwardsamelevel:
     goto backwardsamelevel;
  x = 'wrong';
  return x;
}

function testjumpupward()
```

```
{
  var level = 'test';

  {
    var level = 'top';

    {{
      var level = 'middle';

      {{
        var level = 'deep';

        goto jumpupward;
      }}

      jumpupward:

      if (level == 'middle')
        return 'correct';
    }}

  }
  return 'wrong';
}

function testwronggoto()
{
   try {
     goto nowhere;
   }
   catch(var e) {
      return trim(e).substitute("^.*\\\\", "");
   }
}

function testfunctionwithlabel()
{
  print("before");
  labelinfunction:
  print("after");
}

function testgotointofunction()
{
   try {
     goto labelinfunction;
   }
   catch(var e) {
      return trim(e).substitute("^.*\\\\", "");
   }
}

function testjumpdownward()
{
  try {
    var level = 'test';

    goto downward;
```

```
level = 'top';
     {{{{
     var newvariables= 'new';
     downward:
        return 'wrong';
     }}}}
   }
   catch(var e)
   {
     return e.substitute("^.*\\\\", ""); // warning: the pattern must be ^.*\\  and since it is in a literal we must
write it "^.*\\\\"
   }
   return level;
}

function testdoublelabel()
{
   try {
     var level = 'test';
     goto doublelabel;

     level = 'afterfirstgoto';

     doublelabel:
     level = 'firstgoto';
     return 'wrong1';

     doublelabel:
     level = 'secondgoto';
     return 'wrong2';
   }
   catch(var e)
   {
     return e.substitute("^.*\\\\", "");
   }
}

function testgotoloopforever()
{
  var n = 0;
  try {
     start:
       n++;
     goto start;
  }
  catch(var e)
  {
     return e.substitute("^.*\\\\", "");
  }
}

unittest_areequal(testjumpforward(),'correct', "forward jump");
unittest_areequal(testjumpbackward(), 'correct', "backward jump");
unittest_areequal(testjumpupward(), 'correct', "upward jump");
unittest_areequal(testwronggoto(),'goto label not found: nowhere',"goto label not found");
unittest_areequal(testgotointofunction(), 'goto to label not allowed: labelinfunction', "goto into function");
unittest_areequal(testjumpdownward(), 'goto to label not allowed: downward', "goto label downward");
unittest_areequal(testdoublelabel(), 'label is defined twice in same scope: doublelabel', "label used
```

```
twice");
unittest_areequal(testgotoloopforever(),'goto loop never ending',"goto loop forever");

unittest_areequal(unittest_count(), 8, "incomplete");
print(unittest_results());

================================================================================
============

no output if all tests successful (exit code 0)
error lines if any test fails
e.g.
unit test incomplete failed:  result=7  correct=8

testall.gnscript
  starts internal unit tests and all tests in subfolders
  fullprograms contains many example files that test itself with unit tests
  unit test 100-createindex.gnscript reads all the gnscript examples and creates index.txt that shows
which gnscript features (e.g. functions, operators) are used in the example scripts

========================================
GNSCRIPT USAGE
========================================
gnscript 1.0 by G. Nagler (c) 2021 GN Midi Solutions
usage:
gnscript [options] -tests=[testnamefilter]
gnscript [options] -script text
gnscript [options] filename [params]
options:
-source       show source code
-lines        show all breakable lines
-scan         show scanner token list
-parse        show parser tree
-check        check if parser errors occur
-debug        use console debugger
-lang=xx      use language translations (en, de) for translatable text
-profile=xxx  print times and calling counts sorted ascending or descending
            by method +xxx or -xxx
            xxx can be: count,sumbrutto,sumnetto,singlebrutto,singlenetto

example command lines:
  > gnscript example.gnscript

  > gnscript -script "3+pow(2,3)"
  11

  > gnscript -d -script "function x(n) { return pow(2,n); } return 3+x(3)"
  at console L001C036 return 3+x(3)
  >> s
  at console L001C045 x(3)
  >> s
  at console L001C017 return pow(2,n)
  >> s
  at console L001C024 pow(2,n)
  >> s
  11

  > gnscript -tests
```

951 unit tests successful

> gnscript -d example.gnscript

> gnscript -profile example.gnscript

> gnscript -script "var l='gnscriptdbhash';uselibrary(l); l.getlibrarysummary()'"

==========
BUG REPORT
==========
Please create a small .gnscript example that contains a unit test that fails on your side. It should make clear what result you expected.

example bug report in file example_problem.gnscript:

unittest_areequal(wrongfunction('x','y'), "expected result", __CODEPOS__());

unittest_areequal(unittest_count(), 1, "incomplete");
print(unittest_results());

Please send a bug report by e-mail to info@gnmidi.com

3.138 Mp3 Modifications using FFmpeg tool

Some operations can modify Mp3 audio files if the free FFmpeg package has been installed on your computer.
ffmpeg.exe is an open source free GNU commandline tool to convert and modify media files (video, audio, images).

Disclaimer
FFmpeg is no part and no property of GNMIDI product. GNMIDI only calls ffmpeg.exe if a user has installed the package self.
FFmpeg is a trademark of Fabrice Bellard the initiator of the FFmpeg project (https://www.ffmpeg.org)

Download and Installation

Windows 32 bit (also works in Windows 64 bit systems)

https://github.com/sudo-nautilus/FFmpeg-Builds-Win32/releases

Windows 64 bit

https://ffmpeg.org/download.html#build-windows

Hint: On a Windows computer you can find out if your processor is 64 bit or 32 bit by pressing the key combination WINDOWSLOGO+Pause.

Unzip the zip file using file explorer (or your favorite unzip tool) into a new folder. Make sure that the folders contained in the zip archive are created (ffmpeg\bin).

Start GNMIDI and open a mp3 file and choose one of the available operations (e.g. check file, copy part, fade in, volume) that are enabled in the menus. GNMIDI will tell about required FFmpeg package installation and ask you to select the ffmpeg.exe file with a open file dialog. It should be in the folder where you have unzipped the package inside the ffmpeg\bin sub folder.

After checking if this is a usable FFmpeg version the operation starts.

Flickering
Every time when ffmpeg.exe is started a new (black) commandline window will be visible. Momentarily this cannot be suppressed by GNMIDI.

Speed
Analysing and modifying *.mp3 files costs more time than modifying MIDI files.
mp3 does not store the exact song duration in a header so sometimes it requires to read a file twice (first analyse the file and then modify the file).

Hint: GNMIDI mp3 operations do not overwrite your file. Single conversion creates a temporary result file and batch operation creates results in a result folder.
Hint: do not overwrite your original files without keeping a backup of the original file.

Lyrics in mp3 files
FFmpeg for unknown reasons does not keep the important lyrics contained in an mp3 song (ID3 USLT, SYLT, Lyrics3). GNMIDI tries to copy the original lyrics from input file after using ffmpeg.exe

Problems during conversions
GNMIDI can only use the results of ffmpeg.exe when modifying a mp3 file. If FFmpeg fails for any reason or produces something that is not optimal then GNMIDI cannot create something better.

gnmediahelper

See our product gnmediahelper that helps (batch-) converting or modifying video, audio, images using FFmpeg.

3.139 Favorites

[in menu File]

Favorites
The favorites list collects some files (often MIDI or MP3 files) that are often used by the user. It should be quicklier to find and open those files than using File open.

Toggle favorites state for current document file
The toolbar shows the button FAV deeper if the current document file is in the favorites list and higher if not.
Temporary result files (new*.*) can not be added to favorites before storing them using file/Save as.
Clicking on the FAV toolbar button (or using PLUS short key) adds current file path to the favorites list. If the file was already in the file list the user will be asked if the file should be removed from the list.

Open Favorites (F3)
The dialog displays a sorted list with files currently in the favorites list. Enter a search pattern to reduce the displayed entries matching the search pattern.
The search pattern can be a word sub part (e.g. love) or a sub part wildcard expression (e.g. we*love). The characters match lower and upper characters.
Search priorities:
1. try to match only filenames without extension and without folder path (e.g. Hunting High And Low)
2. try to match filenames including extension without folder path (e.g. Hunting High And Low.mp3)

3. match full filename including path (e.g. c:\music\A-ha\Hunting High And Low.mp3)

Hint: The rules with lower priority are only used if the pattern does not match any entry with a higher priority rule (e.g. a pattern will not be searched in folder path if the pattern is already found in any song file name).

Select one or some entries and use OK button to open them.
Warning: Opening too many files at once is not recommended!

Add Favorites
In open favorites dialog the button can be used to add more files using a file dialog to favorites. The favorites list file will be updated immediately.

Remove Favorites
select one or more items from the list and use button Remove selected Favorites to remove the entries from the favorites list. The favorites list file will be updated immediately.

3.140 Channel range list

Some dialogs offer an edit field for channel numbers 1-16.

This allows specifying
- a single channel number e.g. 10
- some channel numbers e.g. 1,3,5, 10
- a range of numbers e.g. 1-16
- a combination e.g. 1-9,15,16

3.141 Merge lyric tracks

[menu modify]

Some players with lyrics display only use META lyric words that are found in one track. Lyrics could be stored in format 1 MIDI files in more tracks and should be read from all tracks.
This operation collects the META lyric events and moves them to one track.

Hint: Text chords of format [Cm] are not moved to the new track.

Hint: If the MIDI file does not contain lyrics on different tracks then no output is generated.

Hint: In batch operation only files are generated where the input file contained lyrics on different tracks.

3.142 Future operations

New utilities might be added from time to time. Check out one of the following websites for new program versions
- https://www.gnmidi.com

Send ideas for future operations to info@gnmidi.com (it can't be guaranteed that the ideas will be implemented in GNMIDI software).

Index

- . -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- W -

The GNMIDI software may be tested free for 14 days. Download the demo from http://www.gnmidi.com . We hope that you will enjoy GNMIDI as much as many satisfied users have told us yet. If you like GNMIDI please purchase a GNMIDI license at http://www.gnmidi.com/gnorderen.htm

Günter Nagler,
GNMIDI author