converter

release 1.4

user documentation

documentation version 1.9

limitation and disclaimer of warranties

By downloading and/or using converter (herein referred to as "the software" or "converter") and/or the manual (this document inclusive) accompanying this license agreement, you are hereby agreeing to the following terms and conditions:

The software and related written materials (including any instructions or suggestions for use) are provided on an "AS IS" basis, without warranty of any kind, express or implied. This disclaimer of warranty expressly includes, but is not limited to, any implied warranties of merchantability, noninfringement, and/or fitness for a particular purpose. Also, urr Sound Technologies Inc. (urr) does not warrant that the functions contained in the software will meet your requirements, or that the operation of the software will be uninterrupted or error-free or that defects in the software will be corrected. Furthermore, urr does not warrant, guarantee, or make any representations or guarantees regarding the use, or the results of use, of the software or written materials in terms of correctness, accuracy, reliability, current-ness, or otherwise. The entire risk as to the results and performance of the software is assumed by you. No oral or written information or advice given by urr, its dealers, distributors, agents or employees shall create a warranty or in any way increase the scope of this warranty and you may not rely on any such information or advice. You may have other rights which vary from state to state or from province to province. Neither urr nor anyone else who has been involved in the creation, production or delivery of this product shall be liable for any direct, indirect, consequential or incidental damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use or inability to use such product even if urr has been advised of the possibility of such damages. You are not permitted to modify, translate, reverse engineer, de-compile, or create derivative works based on the software, or remove any proprietary notices or labels on the software, without express written permission from urr. This agreement shall be construed in accordance with and governed by the laws of the Province of Nova Scotia, in the country of Canada.

You may freely distribute copies of **converter** provided that the following conditions are met:

- **This license agreement (and hence this manual) must accompany the software.**
- All component files of this software package are distributed together – no files are removed.
- You do not permit other individuals to modify, translate, reverse engineer, de-compile, or create derivative works based on the software, or modify or remove any proprietary notices or labels on the software.
- The software may not be distributed for money or other consideration, or bundled or packaged with another commercial software or hardware product, without express written permission from urr.
- The software is not exploited commercially for purposes other than music production and related applications without express written permission from urr.

Copyright

converter is owned by urr Sound Technologies Inc. and is protected by the copyright laws of Canada and international copyright treaties.

General Provisions

Should any provision of this license be held to be void, invalid or unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby. If any provision is deemed to be unenforceable, you agree to a modification of such provision to provide for enforcement of the provision's intent, to the extent permitted by applicable law. Failure of a party to enforce any provision of this License shall not constitute or be construed as a waiver of such provision or of the right to enforce such provision. If you fail to comply with any term of this agreement, your license is automatically terminated.

new in version 1.4

The following features are new in this release of converter:

- Additional control functions for the 'function' operand in the midi processor and the control functions in the gameport and mouse input subsystems. The new parameters are the audio input filter and gate parameters, enabling quick and easy programming of converter's audio input subsystem via either a midi control surface, custom (knob-based) gameport controller, or mouse input device. Requires use of the control function assignments: refer to the related sections on 'Control Function' in this manual.
- Midi transmission for individual mouse and joystick control channels can now be independently disabled if exclusive 'Control Function' is desired for the particular controls
- Audition key for audio to midi trigger mode programming – pressing the [tab] key while the menu cursor is over a particular audio channel's midi data parameters in the menu system enables the specified sound (if the particular audio channel is programmed to trigger a midi note) to be triggered on the receiving device, speeding up programming tasks
- Metronome 'click track' can be generated either from converter's internal clock or synchronized to an external clock source, with selectable click note number and midi channel – useful for live performance situations where a drummer or other human player requires a monitor mix with tempo cues in order to follow pre-sequenced or otherwise clock-synchronized electronic music / audio
- A few more optimizations here and there.

Bugs fixed in version 1.4:

- Internal midi clock generator completely overhauled and re-written to fix drift issues and stability under various circumstances
- A logic issue in the gameport scanning code which would cause occasional mis-readings of the joystick position has been fixed.
- Overall reliability for slower machines (486 and very slow (ie. 60MHz) Pentiums) attempting large work loads improved
- Joystick input boundary issues from certain calibrations fixed
- Midi processor 'unprocessed echo' of system exclusive messages causing crashes fixed
- Midi input-only mode for SoundBlaster 1.x/2.x/Pro standard significantly improved
- Occasional bootup initialization problems with certain 100% compatible Roland MPU-401 clones (such as Music Quest cards) when used in a dual card input mode fixed
- Gameport and mouse buttons no longer generate double messages (high and low status) when programmed to generate real-time (start, stop, etc) and program change midi messages
- Parameter display translation (float, function, modulator) in incorrect menu pages fixed
- Various minor display bugs fixed (trivial)

As always, if upgrading from an earlier version of converter and placing the new version in the same folder or on the same disk as the older version, make sure to overwrite all the existing files with the new files in the version 1.4 archive. Most of the files are changed in each update, and converter will behave unpredictably if only the main executable (c.exe) is copied, as it relies heavily on its external files.

new in version 1.3

The following features were introduced in version 1.3 of converter:

- Program-specific assignable menu-page bookmarking – assign any menu page to one of eight direct-access user hotkeys ([F1] – [F8]) for immediate access to your most frequently used parameters for a particular program.
- Ability to trigger / control functions in converter from remote midi control using the 'function' operand in the midi processor. For example, use a damper pedal as tap tempo source for converter's midi clock generator, or to turn on and off the midi clock generator itself. Over 45 functions are supported.
- Ability to use any components from the mouse or joystick inputs to control functions in converter using the 'control function' menu / parameters in these particular input subsystems, in much the same way as midi input mentioned above. All functions available to the midi input subsystem are also available to the mouse and gameport subsystems as well.
- Ability to use audio to midi trigger mode using the original single-message type triggering ('normal' mode) or the new 'note on/off' message mode, which generates discrete note off messages in order to support midi equipment or software which does not implement the full midi specification (treating note on velocity zero as a note off messages).
- A few subtle optimizations here and there.

Bugs fixed in version 1.3:

- Patch change decrement bug when view channel is set to [ALL] (oops)
- Tap tempo to ms conversion chart overwriting menu page fixed (trivial)
- Improper redraw when in note view panel during directload bug fixed (trivial)

new in version 1.2

The following features were introduced in version 1.2 of converter:

- Dual interface card support implemented – this provides solid and reliable simultaneous audio and midi input to converter, by using one soundcard for audio input, and another soundcard with an MPU-401 Uart port for midi input. This enables simultaneous audio and midi input for every soundcard standard supported by converter.
- SoundBlaster Pro driver has been improved and perfected for faster computers.
- Some software work-arounds to handle certain hardware problems found on specific soundcards has been implemented.

new in version 1.1

The following features were introduced in version 1.1 of converter:

- Simultaneous midi and audio input (as well as the usual joystick and mouse inputs) now implemented for SoundBlaster 16 (or 100% compatible) users – converter can now do everything, all the time. (NOTE: See section on 'dual card mode support [nov 11/03])
- Support for a huge range of soundcards via implementation of the SoundBlaster Pro audio and original SoundBlaster (non-MPU) midi standard, also enabling converter to be used with many laptops; as an added measure, converter supports a combination of SoundBlaster Pro audio input with separate MPU-401 UART-mode interface for midi output, which may also be useful for certain laptop computers.
- Main audio inputs now with optional band-pass or peaking EQ filters; enables ability to isolate mid frequencies for audio to midi conversion – for instance, snare drum in a drum loop.
- Sum to mono mode for stereo audio input
- All low-pass and high-pass filter channels can use either the original (but improved) shelving filter type, or the new band pass and peaking EQ filter types. Since every audio channel now can have a programmable filter transforming the data, as well as the ability to combine both left and right input channels into a single mono source, it is possible to create midi streams from a 6 band (filter bank) spectra of the audio input source.
- Gain (amplification) for all audio channels
- Beatfinder – an optional visual tool in the hardware and system settings display panel assists in the setting of filter and gate parameters for rhythmically-based audio to midi conversion
- Midi clock tap tempo (insert key) supporting fractional bpm resolution (ie. 120.3 bpm), theoretically unlimited bpm speeds (unlimited), and direct start/stop key (del); greatly redesigned/improved/stabilized midi clock generator; direct hotkeys to adjust clock tempo directly (<ctrl> - <up and down arrow keys>)
- Further optimized for off-the-shelf top-notch real-time performance – 3 millisecond audio to midi conversion latency with default settings (setting can be adjusted easily for even lower latency / faster response), and just a few hundred microseconds average latency on midi I/O processing (essentially, as fast as it gets – basically the limiting factor is the speed of the midi interface standard itself)
- Significantly improved gameport performance; dynamically-adaptive scanning implemented
- Even stronger overload protection implemented
- Directload keys – assign commonly used program files to a range of numerical hotkeys for quick system program change without having to look for a file – great for live use.
- Divide and multiplication operators now use 1/10 float resolution for all arithmetic processors, no longer just within midi input processor – and the menu display indicates value properly
- New 'invert' arithmetic operator for audio, mouse, gameport, and midi input processing
- Midi input processor can now manipulate midi messages up to 100 bytes in length, providing superior transformation support for system exclusive messages
- Increased ranges for many parameters
- More preset (.set) files for a wide range of applications
- Midi processor library file (.set) containing individual operations which may be copied and pasted for simple and quick customized solutions to midi processing applications
- Long program description titles
- New audio settings display panel, directload hotkey assignments display panel
- Much improved oscilloscope display panel, with freezeframe key
- Enhanced midi message scrolling textbox
- Numerous user-interface enhancements and fixes to greatly improve ease of use
- Backwards compatible with version 1.0 .set files; version 1.1 .set files can also be used in

converter 1.0 (however, any version 1.1-specific programming will not be loaded or saved in the old version 1.0 of converter)
- New 'brushed aluminum' look (can be disabled if desired), various color schemes
- Additional keyboard shortcuts

Bugs fixed in converter 1.1:

- crashes caused by midi input while in SoundBlaster audio-only mode fixed
- issues with program change messages generated by [ctrl] [+] and [ctrl] [-] fixed
- division operator in midi input processor fixed (oops)
- audio/gameport/mouse to system common message running status improved and fixed; any other running status issues corrected
- quirks in the load file menu page fixed; quirks in save file menu page fixed, now idiot-proof
- quirks in numerical translation within midi input processor transform operator menu fixed; enhanced (floating point) translation implemented
- midi input processor bypass bug fixed
- all performance quirks with the low pass audio filters fixed; low pass filters can now be set from 20Hz to 5kHz; triggering more accurate; works beautifully.
- not really a 'bug fix' but: the panic button (spacebar) now restores specific controllers to a more useful state – helpful for audio to midi conversion applications in continuous controller mode. For example, panic now resets channel volume to max volume, balance and pan to middle value, pitch to middle value, etc. A couple other quirks (under certain situations) fixed.
- various other incidental (trivial) bugs corrected

As it stands, we are not aware of a single bug left in the code of converter, and every possible effort has been made in development and aggressive testing to ensure there aren't any new bugs that have arisen from the addition of the new features. However, with any software application as complex and flexible as converter, there are always possibilities that a bug exists which may make itself known under certain usage applications that we could not anticipate. Please let us know if you discover one.

For returning users: Many things have changed since version 1.0 of converter. Make sure to re-visit **at least** the sections in this manual regarding the audio menus, user interface & keyboard layout, real-time and display settings menus, software configuration, supported soundcards / hardware, and LFO generators.

contents

introduction to converter

foreword	9
features and uses	11
system requirements	12
supported soundcards / system hardware	13
file listing of the converter package	17

getting started and using converter

dual cards for simultaneous audio and midi	18
software installation and configuration	19
cable connections in a studio environment	21
loading / running converter	22
calibrating hardware settings within converter	23
using converter	24
keyboard layout and hotkeys quick reference	30
midi clock generator	32
directload keys	33
bookmark hotkeys	34

root menus

disk i/o	36
core realtime engine settings	37
auxiliary realtime engine settings	39
display settings	40
customize user panel	42

audio input

system overview	43
gate settings	46
filter settings	47
midi data settings	48
midi data reduction settings	49
arithmetic operators	50

midi input

system overview	51
midi processor program settings	54
midi program operation parameters	55
operation byte check parameters	56
operation transform parameters	57

gameport input

system overview	60
input settings	63
axis midi conversion settings	64
button midi conversion settings	65
arithmetic operators	66
gameport control functions	67

mouse input

system overview	68
input settings	69
axis midi conversion settings	70
button midi conversion settings	71
arithmetic operators	72
mouse control functions	73

lfo generators

system overview	74
transmit enable / disable	76
midi conversion settings	77
arithmetic operators	78

appendixes

signal paths (block diagrams)	79
preset program files	81
midi data protocol specifications	82
troubleshooting	85
proper soundcard configuration under DOS	90
creating a boot floppy to auto-run converter	93
tips to ensure best performance	94
gameport interface pinout	95

foreword

This software was originally created as an in-house prototype tool for direct midi datastream control and conversion of non-midi controller sources. For reasons of flexibility, certain parameters and functionality within the software has remained open in an attempt to avoid limitations in its potential applications; as a result, it is possible for converter to be programmed in ways which do not conform to conventional midi specifications. While the user interface has been carefully designed so that a lot of the functionality of converter can be easily and immediately utilized by someone with minimal experience using midi, **a certain amount of knowledge and understanding of the underlying concepts of midi as well as its lower-level implementation is required in order to take full advantage of the type of control and power that converter provides.** To learn some basics about the midi data protocol specification and related concepts, refer to the midi format appendix. To truly understand converter and how to make it perform to its best abilities, **read this manual carefully** – there are a lot of important pieces of information in various places of this document (many not so obvious).

Even though an experimental software program, **when configured properly converter provides extremely reliable and robust performance**. The nature of its user interface tends to avoid possibilities for crashes or other errors. Any functionality problems will most likely be a result of user programming errors or improperly configured interfaces. However, as the PC world represents denizens of manufactures of every type of hardware component, there are some potential caveats – refer to the troubleshooting appendix if you encounter technical difficulties.

Why DOS?? Why not Windows?!?

Several reasons here. This type of application, with the low latencies it provides, should not require a fast Pentium-3 or Pentium-4 system (or even a Pentium-2 based computer), **which would be required for equivalent real-time performance as a Windows application**. A high-performance soundcard is also not required for the type of work converter performs. The DOS operating system was chosen for its combination of reliability, efficiency, and elegance. The low operating overhead of DOS enables real-time performance comparable to that of dedicated outboard hardware implementations, a requirement for practical applications in live performance and musical interactivity. **DOS can provide this level of performance with much lower hardware requirements than an equivalent Windows application**, and therefore provides an effective utility for older 'obsolete' computing hardware eclipsed by current trends in operating systems and software technology. While consumer soundcards (and especially older soundcards such as the SoundBlaster 16 etc) provide audio quality which is substandard for professional applications in recording, **converter provides an application in which these older abandoned soundcards and computers can find a new life within the context of current professional studio work – thus avoiding the garbage dumps, which is increasingly becoming an environmental problem**. Considering a Pentium-233MHz machine can currently be purchased for about $50 CDN (late 2003), and slower computers (still supported by converter) can be found essentially for free, older computers make great cheap "embedded systems" or dedicated machines for a multitude of purposes.

Attempting to run converter under Windows

During testing it was found that converter behaved unpredictably when run from within a DOS shell under Windows, due to the way it directly programs most of the computer's hardware. Therefore, converter must be executed in "true" DOS mode (without Windows operating in the background). WindowsXP might 'think' it can run converter in a DOS-emulated session – and perhaps it can, under the right hardware conditions – however this is highly unlikely, and will likely result in a crash or harsh corruption (non-permanent) of video card ram (display memory) which is only corrected by a reboot.

A note about the graphics system

The graphics routines in converter are a **low priority** task, since handling the midi data stream and the related input conversion and processing is the most important task the computer has to do. This is by no means to say screen updates are at all slow (in fact you will likely be surprised by the speed of converter's graphics), but as a result, under heavy processing loads or on slower systems the response time of the graphics system may create the visual appearance of a slight computational lag. This doesn't correspond to a lag associated with the actual midi stream. With regards to audio conversion using smaller buffers, often a couple buffers or more will pass by and be processed before the computer is able to update the screen, resulting in the visual effect of the peak meter "floaties" bouncing up from an audio peak that wasn't reflected by the actual peak meter itself (the slower the computer, the more noticeable this effect). Keep in mind that updating the screen is at the bottom of the list in terms of computing importance, and by the time the screen reflects the 'current' input state, the processed or generated midi data has already been transmitted and received by the intended outboard device.

Also note that converter has been programmed in such a way that if the system has a heavy processing load, some of the graphics functions (such as the midi text box) may automatically update slowly or cease to be updated; this is simply converter 'adjusting itself' to ensure the best possible real-time performance on your system.

A note about simultaneous audio and midi input sources

Aside from the LFO generators, there are four sources of actual input within converter – midi, audio, gameport (joystick), and mouse. When configured to use a single soundcard, there are two core input modes in which converter functions – midi input mode, and audio input mode. With the exception of the SoundBlaster AWE64, simultaneous audio and midi input in converter requires a second card with an MPU-401 UART for solid, reliable, error-free real-time performance. This dual-card approach is a result of hardware limitations inherent in these older soundcards. Note that mouse and gameport input (as well as the LFO generators) are always available in all core input modes (both midi input mode and audio input mode). The desired core input mode for converter to operate in (audio or midi) is chosen in the `hardware.cfg` file. This topic is discussed further in the section entitled "software configuration".

For further details, refer to the section in this manual entitled "supported soundcards / hardware".

Keep in touch with us!

If you find converter useful, please let us know what you think of it, any suggestions you might have, what kinds of projects you've used it on, what types of computers / soundcards you are using it with, or any other information and/or suggestions. Thanks to everyone for their feedback so far.

Make sure to visit our website (www.urr.ca) from time to time in case there is an updated version posted (as this release was), or to find other new tools and technologies we are already developing.

If you have problems, or found a bug we should know about, you can reach us at support@urr.ca, or by posting in our user forum (which may already have answers to your questions).

Rest assured that any personal information we receive remains completely confidential, including personal email addresses. We do not sell address lists, and we despise spam.

Thanks for checking out converter! We believe it's a very strong application, and is the result of many thousands of hours of work. Hopefully it at least meets if not exceeds your expectations!

Tom Roscoe
urr Sound Technologies Inc.
www.urr.ca

Main Features

- extremely low-latency system response time similar to equivalent dedicated outboard hardware
- programmable midi input processor allowing for 144 complete midi message operations in real-time, via a total of 432 conditional verifiers and 576 data transforms
- 2 channels of audio input, each with 2 additional filter channels for a total of 6 independent sources for audio to midi conversion; each channel is equipped with individual conversion modes, filters, gates, gain settings, intelligent data thinning algorithms (for continuous controller conversion), and arithmetic processors
- audio channels can be configured as a six-band filter bank for complex spectral based audio to midi conversion
- selectable filter types for each audio channel including shelving, band pass, and peaking EQ, all with individual frequency and Q
- audio can be converted to midi in both trigger/gate as well as continuous controller modes (functions like an envelope follower)
- fully programmable gameport and mouse to midi conversion providing 12 additional channels of variable and binary midi control, allowing for the design and implementation of custom midi control surfaces
- midi clock generator with tap tempo, quick control keys, and auto-sync click track generation
- 8 clock-synchronizable lfo generators, to generate midi continuous controller streams or used as modulation sources for the midi input processor
- highly configurable user interface and real-time performance load – converter can be adapted to perform well on any machine from a 486DX/66 to a Pentium 233MMX and beyond
- supports a variety of laptops for portability in live performance situations
- costs almost nothing for the type of hardware required to operate converter
- very stable – can operate indefinitely without requiring a reboot

Uses

- add additional controller functionality or midi data support to equipment with limited controller programmability or primitive midi interfaces (such as vintage midi synthesizers and drum machines, drum pads, stage lighting controllers, effects processors, etc.)
- clock-synchronized low frequency oscillators can add synchronized sound-shaping features to classic synthesizers – useful in sequencing environments
- convert certain system exclusive messages in real-time; translate from one format to another
- provides new possibilities in sampling by providing a new type of envelope follower, allowing one sound to adopt various rhythmic or dynamic inflections of another sound in real-time (as a basic example, impose a rhythmic pulse to a filter on a sound on a synthesizer, based on a drum loop passed into the audio input in converter)
- enables sounds on an external midi device to be triggered from audio input peaks, for example a mic'd snare drum could trigger a sampled snare sound etc.
- provides quick and easy master midi clock for a midi setup in a live performance application or creative studio environment, with the ability to provide reference tempo cues for human performers
- trigger sound, lighting, or other events from objects on a stage in a live theatrical, musical, or new media performance
- provides a useful tool for debugging or testing midi systems / implementations
- provides relevant and interesting uses for otherwise abandoned legacy (i.e. "obsolete") hardware – material which is classified as 'dangerous goods' or is otherwise non-biodegradable

system requirements

Different aspects of the software require more processing power than others in order to deliver fast response times. Even though conceptually simple, converter must perform a tremendous number of computing instructions per second in order to provide the kind of programming flexibility it allows for the multiple input sources it supports. As a very rough guideline, a 486DX-33 system is recommended as a minimum for midi processing or 2 channel audio conversion, and a Pentium-based PC at 100MHz or greater for full audio support (including all filters enabled). A Pentium 166MHz or greater is recommended as a minimum for using converter in dual card mode with all four input engines operating simultaneously. If you are using a significantly slower machine and performance seems to be impacted for some reason (and it isn't due to a hardware conflict or other issue out of converter's control), there are many configurable options described in this manual that will assist in maximizing the performance of converter (refer to the appropriate appendix). Note that a math coprocessor is required for 486-based computers (which is included on-chip by default on most 486 systems).

Required:

- 486DX-33MHz based PC as an absolute minimum for basic use**, Pentium 100MHz or faster strongly recommended**, Pentium 166MHz to 233MHz suggested for more complex applications
- DOS (probably 4.0 or higher – the DOS with Windows 95 / 98 is more than sufficient)
- SVGA video card with at least 512k (0.5 MB) of video RAM
- 1 MB of system RAM (yes, just 1 megabyte)
- about 560k of free conventional memory (the base 640k that DOS uses – don't worry about this point unless you use DOS regularly and have several memory-resident DOS programs loaded)
- one of the supported sound or midi interface cards or a compatible alternative – see next page.

Ideal machine configuration (suggested) for aggressive processing loads

- Pentium 200MHz (or faster)
- later PCI (ie. ATI Mach64) or early AGP (preferable) (ie. ATI Rage, etc) video card
- SoundBlaster 16 (or AWE32, AWE64, etc.) for audio and joystick input
- Roland MPU-401 or other quality MPU-401 UART mode interface for midi input (this type of midi port can be found on most soundcards)
- Cirque serial or PS/2 Touchpad pointing surface (or similar Logitech/Microsoft compatible device)
- hard disk of whatever size is convenient (40MB would function fine)

supported soundcards / hardware

There are several standards for interface cards which converter supports. These standards are implemented on a variety of cards; however testing for each potentially compatible card is impossible due to the sheer number of interfaces on the market. The following guide outlines the interface standards implemented in converter, the specific interfaces which were used in the development of converter (and hence are directly known to work), and any information about cards which typically wouldn't work.

If you are successful using an interface card that is not listed here as specifically known to work, or you find a card that should work but doesn't for some reason, please let us know so it can be added to this list and assist other users in the future.

Note that soundcards will require a midi/joystick breakout cable or breakout box in order to be connected with standard midi cables. These are standard breakout cables or boxes which should work across all the different types of consumer soundcards on the PC market (including modern PCI soundcards), and can still be purchased or built (see www.harmony-central.com/MIDI/interface.html). If purchasing, simply look or search for a "SoundBlaster midi breakout cable" or "SoundBlaster midi adaptor". A variety of companies (such as Midiman) manufacture very inexpensive solutions – it certainly doesn't have to be 'Creative Labs' or 'SoundBlaster' - branded. The better ones are opto-isolated, to protect your computer and outboard midi equipment, and have an LED on the midi input and output ports to indicate midi transmission activity.

Roland / Generic UART-mode MPU-401

DEVELOPMENT INTERFACES: ROLAND MPU-IPC-T, MUSIC QUEST MPU-401 CLONE, SOUNDBLASTER 16 SERIES, SOUNDBLASTER AWE-32/AWE64 SERIES, ENSONIQ SOUNDSCAPE

Any 100% MPU-401 UART-mode hardware compatible midi interface card is supported by converter. This includes the Roland family of interfaces (MPU-401, MPU-IPC-T, LAPC-1, SCC-1, etc.), and should include various devices from MusicQuest, Turtle Beach, Terratec, and hundreds of others. Note that the 'intelligent' mode of the MPU-401 is **not** used – only the UART mode.

Users with a SoundBlaster 16 (or newer) card should use the SoundBlaster midi mode of converter, since it will automatically configure itself for your SoundBlaster's MPU-401 interface based on the SET BLASTER line in your autoexec.bat file.

Gravis Ultrasound

DEVELOPMENT INTERFACES: ULTRASOUND 'CLASSIC' (GF1-BASED)

There are several variations of the Gravis Ultrasound ("gus") card based on two different main processor chips. The first series of cards were based upon the GF1 processor, and as such are: Ultrasound ('classic'), Ultrasound Max, Ultrasound Ace (**not** recommended), and Ultrasound Extreme. The more recent Ultrasound cards, called either the Ultrasound Plug & Play or Ultrasound Plug & Play Pro, are based upon the AMD Interwave chip. Either the Ultrasound 'classic' or Ultrasound Max will work with converter (with the SET ULTRASND line in your autoexec.bat file set correctly – the installation software should do this automatically). The Ultrasound Extreme and Ultrasound P&P series *may* work (we have no way of testing these cards), while the Ultrasound Ace will **–not-** work (it has no ability to record audio, and no hardware midi interface).

SoundBlaster Cards

There are a tremendous number of variations of the SoundBlaster family of soundcards. Early SoundBlaster cards (SoundBlaster 1.x, SoundBlaster 2.0) are supported for midi input, but not audio input, since they cannot support even the most basic audio requirements of converter. Newer PCI-based SoundBlaster cards (SoundBlaster pci-128, SoundBlaster pci-512, SoundBlaster Live!) are **not** supported since they do not have complete SoundBlaster 16 or Pro compatibility – it is provided through a memory-resident driver which doesn't support midi or audio input. **In general, it is safest to go with an ISA-bus card for true DOS software compatibility**; some PCI-bus cards (perhaps Terratec?) *may* provide a memory-resident SB16 compatibility driver that does implement emulated hardware-level midi and audio input – please let us know if you have success with one.

Even though supported by converter, we advise against using the old SoundBlaster Pro / 1.x / 2.x midi interface standard for midi input and output, as it was a problematic interface standard. Historically, users have had problems with these older cards for anything more than light to medium midi stream density, and while problems should be relatively rare, reliable results can not be guaranteed for critical applications. If possible, use an MPU-401 interface instead.

SoundBlaster 16 (or newer)

DEVELOPMENT INTERFACES: SB-16, AWE-64, AWE-32 (all ISA-bus)

Cards known to work are:

- SoundBlaster 16
- SoundBlaster 32
- SoundBlaster AWE-32
- SoundBlaster AWE-64
- SoundBlaster AWE-64 Gold

Cards which should also work (since they are supposed to be 100% hardware or "register-level" compatible with the SoundBlaster 16 standard), *but have not been tested or confirmed*, are:

- Avance Logic ALS-100, Avance Logic ALS-200
- ASound 32/3D PnP, ASound Gold
- Zoltrix AudioPlus 6400, Zoltrix AudioPlus 3200

SoundBlaster Pro (or newer)

DEVELOPMENT INTERFACES: ESS ES1688 AudioDrive (SBPro clone), AWE-64, AWE-32

While fully (and properly) supported and tested, there are some issues with certain motherboards and SoundBlaster Pro cards, namely resulting in lost buffers of audio or other DMA-related issues which are out of our control. These problems do not seem to affect the SoundBlaster 16 standard above, so if you have the option (for example, buying a 2nd hand card), choose a SoundBlaster 16 card instead.

Cards known to work are:

- SoundBlaster Pro, SoundBlaster Pro 2
- ESS ES1688 AudioDrive, ESS ES1788 AudioDrive, etc.

Cards that almost certainly should work since they claim to be 100% hardware compatible with the SB Pro (*however have not been tested or confirmed*) are:

- Media Vision Delux 3D Pro
- Logitech SoundMan Wave
- Terratec Xlerate, Terratec EWS64 XL, Terratec Soundsystem DMX
- Turtle Beach Tropez, Turtle Beach TBS-2000

Some of the literally hundreds of cards which should also work (since they are supposed to either be 100% hardware or "register-level" compatible with the SoundBlaster Pro standard, or have some sort of driver which provides the necessary compatibility) are:

- OPTi MAD16 Pro
- VIA AC-97 Sound
- Acer Magic
- Atlantis CS4232
- AVM Apex
- S3 Sonic Vibes
- Vortex AU8810, Vortex AU8820 AC'97
- Sound Galaxy Pro
- Dream 32GM
- Terratec 128i PCI
- Media Vision Jazz16

Standard Analog Gameport Interface

DEVELOPMENT INTERFACES: GAMEPORT ON GRAVIS ULTRASOUND, SOUNDBLASTER AWE-64, AWE-32

The gameport interface that converter supports is the original IBM-spec standard analog gameport, which was originally available as an add-on card, and eventually became a standard component of soundcards as they started being manufactured. Therefore, if you have a Gravis Ultrasound, SoundBlaster 16 or compatible, or SoundBlaster Pro or compatible soundcard, you also have a standard analog joystick port. Newer types of gameport interfaces, such as advanced force-feedback digital interfaces, are not supported - however these newer advanced types are much less common, and certainly not found on legacy soundcards.

Standard Microsoft or Logitech-compatible Mouse

DEVELOPMENT INTERFACES: SERIAL-PORT CIRQUE GLIDEPOINT TRACKPAD, LOGITECH PS/2 WHEEL MOUSE, LOGITECH PS/2 OPTICAL WHEEL MOUSE, GENERIC SERIAL MOUSE

Any mouse or pointing device which is supported by an appropriate DOS mouse driver is supported by converter, as it interfaces with the DOS driver that either came with the device or is downloadable from the manufacturer's website. Since most manufacturers completely adhere to the Microsoft/Logitech mouse interface standard, most devices should work automatically – basically any sort of pointing device that can be plugged into a PS/2 or serial port should work.

Make sure that you load the mouse device's driver each time you boot into DOS in order for converter to detect the mouse – a good idea is to place the command to run the driver (ie. `mouse`) in your `autoexec.bat` file, so the driver is automatically loaded each time DOS is booted. DOS mouse drivers are relatively easy to find on the Internet – there's always www.logitech.com.

Laptop Support

As many older laptops were manufactured with built-in sound support on their motherboard, usually with SoundBlaster Pro compatibility and perhaps better, many older original Pentium-level laptops can **theoretically** run converter, providing a portable tool which can be used in live performance without having to lug a delicate monitor or large computer around. The main thing to be aware of: **the laptop will need to have either the standard 15-pin joystick/midi breakout connector on its back panel (apparently more common on older higher-end laptops), or some sort of connector (such as on the IBM Thinkpad series) which accepts an adapter cable to provide a 15-pin joystick/midi breakout connector.** Sometimes the 15-pin connector is not on the back panel of the laptop itself, but on the docking station designed for the particular laptop. Note that the 'SoundBlaster Pro audio input / MPU-401 midi output"

mode may be useful for certain laptops. Also, remember to have the “SET BLASTER” environment string in your autoexec.bat file – more information about this is found in this manual’s appendix, in the section entitled “proper soundcard configuration under DOS”.

Video Card Support

DEVELOPMENT INTERFACES: Number9 / S3 PCI, MATROX G200 AGP, ATI MACH-64 PCI, ATI RAGE IIC AGP, ATI RAGE 128 AGP, CIRRUS LOGIC 54xx VL-BUS, ATI RADEON 9700 DUAL HEAD AGP…

Almost any video card is supported, either via its own chipset standard or through VESA compatibility mode. If at first converter doesn’t seem to function (either there is garbage on the screen, or no image at all), try to locate DOS VESA drivers for your card, or try SciTech’s Display Doctor program. In general, any video card found in (or even compatible with) the older computers for which converter was designed should work properly. The very latest AGP 4x or 8x video cards which *may* not be compatible with converter would not even function in an older computer anyway, due to different power standards. Newer AGP video cards (manufactured from about 2002 onwards) are designed for 1.5 volt operation while older AGP 1x and 2x cards were designed for 3.3 volt operation. Using newer video cards in older computers will result in either the video card or the motherboard burning out on power up – so be careful, if assembling machines with a combination of old and new components.

If undesirable visual effects (such as flickering during certain text animations) is a problem for your particular video card, simply disable the animations in the Display Settings menu ([Alt] – [D]).

Certain video chipsets, particularly ones which require support through VESA compatibility such as certain on-board video chipsets (such as those found in pre-fab PCs such as Dell PCs), will provide less than optimal (ie. slower) graphics performance; however converter’s graphics performance will certainly still be completely usable.

Flat Panel Display Support

As flat panel display technology continues to leap forward, older 15” analog-compatible (15-pin sub-D connector) flat panel (LCD) displays which can only support 640x480 or 800x600 resolutions are being tossed aside for a small amount of money. These displays work perfectly with converter, which uses a screen resolution of 640x480. Yeah, believe it or not, we’ve been asked this question!

file listing of the converter package

The following is a listing of all the files included in the converter source zip archive, and their relationship with the main program file. Files noted as required need to be in the same directory (folder) as the converter program file (`c.exe`) in order for it to operate properly. Files noted as optional are not needed for proper operation of converter, but are highly recommended to be kept together.

file	description	dependancy
`c.exe`	converter program file (executable)	required
`c.cfg`	converter system configuration file	required
`hardware.cfg`	soundcard interface settings	required
`c1.cdt`	converter data file	required
`c2.cdt`	converter data file	required
`c3.cdt`	converter data file	required
`c4.cdt`	converter data file	required
`c5.cdt`	converter data file	required
`cp.urr`	converter data file	required
`cb.urr`	converter data file	required
`cp0.urr`	converter data file	required
`cp1.urr`	converter data file	required
`cp2.urr`	converter data file	required
`cp3.urr`	converter data file	required
`uipa.urr`	converter data file	required
`uipb.urr`	converter data file	required
`default.set`	default program file loaded when converter is initially run	required
`empty.set`	'blank slate' / reset converter to default settings	recommended
`cmanual.pdf`	manual for converter (this document)	optional
`autoexec.bat`	a basic autoexec.bat file which can be used for a bootable floppy disk installation of converter	optional
`lastset.set`	'revert to previous' setting file	optional
`m-opslib.set`	preset : midi input processor operations library	optional
`gpmixcut.set`	preset : gameport input as midi mixer / mute (cut)	optional
`dltrigX.set`	preset : audio to midi trigger program files	optional

There are three external files used by converter which serve different purposes – all three of these must be located in the same directory (folder) as the executable file (c.exe). The first file, `hardware.cfg`, has already been discussed as it is a text file which holds the individual computer-specific hardware settings for the soundcard used. The file `default.set` is the base program file which is read from when converter is initially run, and updated (written to) when converter is quit. The final file, called `c.cfg`, is read from at run-time and written to on exit, and is used to store other computer-specific hardware settings adjustable from within converter. In this way, programmed converter configurations (`.set` files) can be interchanged with other users, while machine-specific settings remain tied to the particular machine on which converter is being used. At the same time, `c.cfg` and `default.set` provide something akin to a 'battery-backed memory' so that the last converter session, with all its settings, is automatically re-loaded and ready to be used the next time converter is run.

parameter settings saved exclusively in c.cfg

- display settings (vertical retrace, animation settings, oscilloscope settings, scrolling text window status, midi input processor view mode, etc.)
- input device settings (joystick axis channel calibrations, mouse axis speed)

dual cards for simultaneous audio and midi input

Since many consumer soundcards (notably the SoundBlaster 16 and Awe-32s) have difficulties handling dense midi input streams in combination with 'high speed' audio input or output, support for dual soundcards has been implemented in converter in order to provide trustworthy and reliable performance.

When using one of the dual card modes of converter, audio input is handled by one soundcard, and midi input and output is handled by a separate card with an MPU-401 UART interface. The hardware settings for this secondary MPU-401 card for midi I/O are determined using the last 2 parameters in the `hardware.cfg` file (described in the next few pages of this manual).

This dual card support is very flexible, in that a variety of combinations of soundcards could be used to achieve solid performance. Two SoundBlaster 16s, an Awe-32 and an ESS1688, a SoundBlaster Pro and a Roland SCC-1, the options are numerous.

For optimal results, it is recommended that the MPU-401 interface used for midi input and output has a lower interrupt setting than the card used for audio input. For example, if using a Roland MPU-IPC-T, the card's default settings of IRQ 2 would be optimal when used in combination with a SoundBlaster 16 set to it's default IRQ of 5 for audio input.

Also, it is recommended (in general) to use a quality MPU-401 interface for midi input and output, if possible. As an example, while the MPU-401 ports on most SoundBlaster cards work fine, there are issues with certain older revision Awe-32 and some other cards which occasionally lose midi data during dense incoming streams. If you have an alternate card with an MPU-401 interface, such as a Roland card, Ensoniq Soundscape, Turtle Beach etc., use the Awe-32 for audio input and the other card for solid midi input and output functionality.

Configuring two Plug & Play SoundBlaster cards in the same machine may be challenging in some computers; these cards may require that the "Plug & Play OS" parameter in your computer's BIOS be set to "OFF" in order to make use of the soundcard's hardware configuration software. Usually it is easiest if Plug & Play cards have their settings assigned from within Windows, and other times it is easier when one card's hardware settings can be set via jumpers found on the soundcard itself.

The SoundBlaster AWE-64 is able to handle both input streams on one card, so if you are using this particular card, you can use input mode 5. Other better quality soundcards, such as the Turtle Beach Tropez or EWS-64 may (very likely) be able to perform a dual card mode using just the single card – however we are unable to test this as we do not have these particular cards.

If your soundcard has an MPU-401 port as well as SoundBlaster Pro or 16 audio compatibility, and uses two IRQs (one for audio I/O and one for the MPU port) it is possible to use one of converter's dual card modes on the single card. This will depend largely on the reliability of the card itself (again, older Terratec or Turtle Beach cards may work well in this situation, but again we have no way to confirm this directly).

To 'install' converter, simply create a directory (folder) somewhere on your computer's hard drive, and copy all the files contained in the converter .zip archive to that folder. Or alternatively, copy the files to a blank 1.44MB 3½" floppy disk (see the appendix on creating a bootable floppy).

Before running converter, the file `hardware.cfg` should be examined to ensure the settings are appropriate for your particular soundcard. This can be loaded using any text editor – in Windows you can use Notepad, or under DOS you could use Edit). Make sure not to change the layout of the file (don't delete any lines, even blank ones). The parameter to be adjusted is just below the listing of options, and is **not** preceded by a '#' sign.

If the default settings (listed below) suit your hardware, there is no need to edit this file.

The file `hardware.cfg` looks like this (default settings):

```
# CONVERTER HARDWARE SETTINGS CONFIG FILE
# ------------------------------------------------------------------------
# ======  Do not add text or otherwise change the line spacing of this file.|
# |NOTE|  Just edit the number for the parameter you need to change.        |
# ======  Altering the layout of this file may cause converter to not load. |
# ------------------------------------------------------------------------
# SOUNDCARD TYPE + CORE INPUT MODE
# ____________________________________________________________________________
# MODE | SOUNDCARD                     | INPUT MODE              | CARD MODE |
# ----------------------------------------------------------------------------
# 0    | Roland/UART-MODE MPU-401      | MIDI                    | SINGLE    |
# 1    | Gravis Ultrasound             | MIDI                    | SINGLE    |
# 2    | Gravis Ultrasound             | AUDIO                   | SINGLE    |
# 3    | SoundBlaster 16 or newer      | MIDI                    | SINGLE    |
# 4    | SoundBlaster 16 or newer      | AUDIO                   | SINGLE    |
# 5    | SoundBlaster AWE-64           | AUDIO & MIDI            | SINGLE    |
# 6    | SoundBlaster Pro or clone     | AUDIO                   | SINGLE    |
# 7    | SoundBlaster 1.x, 2.x, Pro    | MIDI                    | SINGLE    |
# 8    | SoundBlaster Pro & MPU-401    | AUDIO                   | DUAL      |
# 9    | SoundBlaster Pro & MPU-401    | AUDIO & MIDI            | DUAL      |
# 10   | SoundBlaster 16(+) & MPU-401  | AUDIO & MIDI            | DUAL      |
# 11   | Gravis Ultrasound & MPU-401   | AUDIO & MIDI            | DUAL      |
# 20   | No Sound Card (video test)    | NO I/O                  | N/A       |
# ----------------------------------------------------------------------------
4

# AUDIO INPUT BUFFER SIZE (in bytes)
# can be no smaller than 12 and no greater than 512; default is 80.
# settings lower than 72 will automatically disable joystick input.
80

# ------------------------------------------------------------------------
# SETTINGS SPECIFIC TO GENERIC UART MPU-401 ONLY,
# OR MPU-401 IN DUAL CARD INPUT MODE.
# ------------------------------------------------------------------------
# MPU-401 IRQ
# possible settings: 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15
2

# MPU-401 BASE ADDRESS
# possible settings: 300, 310, 320, 330, 332, 334, 336, 340, 350, 360, 370, 380
330

# END OF CONFIG FILE.=====================================================
```

The first parameter allows you to select the soundcard and what core input mode you want to use depending on your system's soundcard (audio input or midi input for most of the soundcards, and the additional option of simultaneous audio and midi input for users with an additional MPU-401 card).

The second parameter only affects audio input, and allows you to define the size of the audio buffer. This determines both the response time for audio to midi conversion, as well as the quantity of midi data generated from the audio input (in continuous controller mode). Since each buffer of data is converted into a single amplitude value, the smaller the buffer size, the greater number of times it is converted per second, with lower latency – and conversely, the larger the buffer size, the fewer number of times it is converted, with greater latency. The smallest buffer size you can use is determined by your computer's processing speed; smaller buffer sizes result in significantly more instructions per second and I/O functions that the computer must complete in order to avoid losing audio samples, and will have an impact on the speed of the overall system. **The side effect of using a very small buffer is that the software will actually trace individual cycles of lower-frequency waveforms, resulting in midi continuous controller streams that oscillate at a very fast rate, much like an LFO on a synthesizer**. This ability has been preserved for those wishing to experiment with these types of oscillations as control signals. To avoid this type of effect, try using an input buffer size of 80 bytes or greater – experimentation is key. In general, a buffer size below 80 (and certainly below 64) bytes is unnecessary.

For most applications, the default buffer setting of 80 is fine, and gives you a low latency of ~3 milliseconds.

Note also that with a buffer size less than 72 bytes, gameport input is automatically disabled – refer to the gameport system overview for further details.

The last two parameters are to define the hardware settings (interrupt (IRQ) and base port address) for those using a Roland or generic UART-mode MPU-401 for midi input in mode 0, or midi input & output in one of converter's dual cards input modes. For the SoundBlaster Pro & MPU401 MIDI output mode (mode 8), only the base address of your MPU-401 interface is needed – the IRQ for the MPU-401 is not used by converter in this particular mode. For modes 0, 9, 10, and 11, both the base address and IRQ values are needed for your particular MPU-401 interface. The numbers for the base address and IRQ used by your MPU-401 can be found in the resources tab of its driver under Windows 9x (if you are using Windows 9x) and just entered here. The factory default for an MPU-401 is base address 330 at IRQ 2/9.

If using a dual card mode for converter, it is recommended that the interrupt assigned to the MPU-401 interface which is being used for midi input is lower than the interrupt assigned to the card performing audio input. For further information, see the section in this manual entitled "dual cards for simultaneous audio and midi input".

Once you have customized these settings to your specific needs, save the file and you're ready to run converter.

cable connections in a studio environment

In a medium-complexity studio environment, with a fairly flexible mixing console and several pieces of midi-compatible outboard equipment, converter operates most conveniently if connections to it are done in a similar manner as a hardware effects processor. By using one or two free auxiliary send(s) on the mixer to feed audio signals to converter, a signal from any one of the mixer's channels can be routed to the left or right audio input of converter at any time, without requiring any change in cabling; in addition, the signal level sent to converter can be quickly and conveniently adjusted for optimal tracking and conversion results. The midi input and output connections are best used in conjunction with a midi patchbay, if available, to enable quick midi signal routing to any device in the studio.

By making connections in this manner, additional pre-processing of the audio signal may be performed before it reaches converter, and this may enable greater flexibility in applications such as generating triggered drum hits from an audio signal with several instruments mixed together. While converter provides quite remarkable ability in separating signals through the use of its filters and gates, there are some occasions where further pre-equalization or expansion may provide even better results.

It may take time and a bit of experimenting to achieve the desired audio to midi triggering result, but the necessary tools and ability are definitely embedded in converter – just have patience and take your time.

Unfamiliar with DOS?

If you are using Windows9x on your computer, you can press [F8] just as Windows begins to load in order to bring up a boot menu, from which you can select "command prompt only" to boot into 'pure DOS' ("command prompt only"). Or, you can boot out of Windows9x by selecting 'Shut Down' from the Start menu, and selecting "Restart in MS-DOS Mode".

In general, the best and easiest approach for running converter on a computer running Windows9x or higher is to create a self-booting floppy disk with converter installed on it; using this approach enables converter to be automatically run with its own `autoexec.bat` settings by simply inserting the floppy and booting (or re-booting) your computer. See the appendix in this manual that discusses this procedure.

If you installed converter on your hard disk: After booting into DOS, change to the directory (folder) you've installed converter by using the cd command, either `cd..` to move back one directory level, or `cd convert` to move into a directory (where convert is the name of the directory). Once in the appropriate directory, run converter by simply typing "`c`" and pressing enter.

Problems?

- If converter fails to load and function properly, make sure to check the troubleshooting appendix of this manual – there may be something about your computer's configuration that has been overlooked. If your problem is not solved there, try posting in the forum at www.urr.ca, or sending an email to support@urr.ca.

calibrating hardware settings within converter

Now that converter is up and running, there are a couple hardware-specific settings found within converter that should be set up so that converter functions predictably and properly. These steps are somewhat optional, as they pertain to joystick input and mouse input; however, if these input sources are to be used, performing the joystick calibration is required, and adjusting the mouse axis speed parameters is highly recommended. These settings are machine-specific, and are not changed when loading different program (.set) files.

joystick calibration

Due to the (somewhat flawed) way in which the standard IBM-PC joystick interface was designed (back in the very early 1980's), there can be substantial differences in the circuitry between various interfaces and joystick/gameport controllers. Therefore, performing the calibration routines for your particular gameport axis channels provides converter with the ability to properly compensate for the ranges of your machine's hardware.

Make sure that the gameport device (joystick, customized controller, etc) to be used with converter is plugged into the gameport connector on the computer, and perform the following steps:

- Press [F4], followed by [F5], and then [F5] again. This is the menu which holds all the configurable parameters for each axis of the two joystick ports (A and B) on the gameport. The first two columns of parameters can be ignored for now.
- On the right hand side of the menu screen, you will notice four pieces of text: "**calibrate joy A-X axis**" (referring to joystick A's X-axis channel), "**calibrate joy A-Y axis**", "**calibrate joy B-X axis**", and "**calibrate joy B-Y axis**". Using the arrow keys, move the menu cursor (the blue rectangle) over to the label entitled "**calibrate joy A-X axis**" and press enter or return.
- Follow the on-screen instructions to calibrate for the axis. After completing the three steps, you will be returned to the joystick input settings menu page. Note that during calibration, all real-time functionality of converter is suspended.
- After completing calibration for joystick A's X-axis channel, repeat the same steps for any additional joystick axis you may be using.

mouse motion speed parameters

In order to ensure full range of motion (or limit the range if preferred), especially for those using touchpad or trackpad mouse controllers, adjust the mouse axis speed parameters as follows:

- Ensure that mouse input is enabled (if this is the first time converter is run, mouse input is on by default). The default assignments for the mouse's X and Y axis are pan and modulation wheel on midi channel 2 by default. Move the mouse; if you do not see the pan or modulation wheel meters move, press [Alt]-[Esc], followed by [F6], move the cursor to "mouse input" and press [page up] to enable the input.
- Press [F4], followed by [F6], and then [F5]. This menu provides four parameters, the bottom two which apply here.
- Using the arrow keys, move the menu cursor to the parameter entitled "**x-axis speed**". While moving the mouse, adjust this parameter to suit your preferences either by using the [page up] or [page down] keys or by typing in a numeric value from 0 to 127.
- Perform the last step for the "**y-axis speed**" parameter.

While the user interface in converter might seem daunting at first, with many pages of menus and parameters, don't be intimidated – there has been a lot of planning behind it to ensure each stage is as intuitive and straight-forward as possible. Most core menu sections have similar (if not identical) layouts, and their patterns will become very obvious after a brief period using converter. Take advantage of the shortcuts and hotkeys, as they will make navigation even faster and easier.

in general

Since converter was designed as a real-time processor, the philosophy behind the display and user interface of converter is that of **maximum real-time visual feedback**. When you first run converter, you will notice a subdivision in the screen layout. The top 2/3rds of the screen consists of a grey, metallic midi message display panel, which is broken down into different 'wedges'. Each wedge groups various graphical objects used to display specific midi messages, or information on converter's activity. Below that, a blue display panel that can be set to display several different pages of system and input information. At the very bottom of the screen is a strip of text displaying three piece of information:

- the current midi channel whose data is displayed in the top section of the screen (when set to [ALL], all data from all or generated for all midi channels is displayed).
- The MTC time count (labeled as 'smpte', since MTC is a sub-protocol in midi to allow a mutated form of smpte to be transmitted over midi
- The midi clock tempo in beats per minute (bpm), and the current count of the clock either received over midi or generated by converter itself.

The top grey/metallic portion of the screen provides real-time display of midi data both generated by converter (from audio and gameport inputs) and received (from a midi input interface). The upper left corner of this panel indicates midi activity on a particular input source, active channels in the output midi stream, and a processor buffer meter which roughly indicates system computing load (based on how heavily the computer is relying on its various processing buffers). Because of the high prioritization of the conversion and midi output systems, a high processing buffer load doesn't normally reflect any additional lag in system response time; however it does give an indication of how hard the computer is working to provide continuous output in addition to the real-time graphics system. For example, if converter is operating in audio input mode with all channels active in continuous controller conversion, the processor buffer will indicate higher buffer usage due to the overall processing load on the system.

The lower display panel can display four different pages of system and input information, accessible by pressing function keys [F9] through [F12]. One or two of these pages may not display any active information depending on the mode in which converter is set to operate (midi or audio input mode). Pages accessed by [F11] and [F12] can be switched between two information pages by pressing [Shift]-[F11] and [Shift]-[F12].

- **midi note messages [F9]** : displays peak meters for all 128 notes, indicating note on and off messages with their note number and key velocity, and indicates whether top and bottom display panels reflect processed or unprocessed input midi data (relevant only in midi input mode). Also provides a simple note number to note name and octave display.
- **audio input oscilloscopes [F10]** : displays waveform plotting (oscilloscopes) for each of the six audio channels, with associated peak meters. This display page takes the most graphics horsepower to maintain – if the redraw speed is slow, try setting the **oscilloscope redraw** parameter to intelligent (in display settings, accessible from the root menu by pressing [F7]). More information on accelerating system performance can be found in an appendix towards the end of this manual.

- **hardware & system settings / audio filter settings [F11]** : when set to display 'hardware and system settings', this display page shows the hardware settings for the selected audio or midi interface card, the data rate of midi transmission (useful to avoid overloading midi bandwidth), the filename for the current settings file and whether or not it has been edited since being saved last, long descriptive name for the current program, input information from the gameport (if enabled), and audio peak meters for each of the six audio channels. If set to display "audio input and filter settings", this display page shows the filter settings (frequency, Q, gain) for each audio channel, the midi conversion settings (conversion mode, target message to convert to) for each audio channel, mouse to midi conversion settings, midi data rate transmission, and audio peak meters for each of the six audio channels. Integrated in the audio peak meters are several visual guides: at the top of each peak meter is an icon representing the filter type (if used) for each channel, and the beatfinder leds. Along the sides of the peak meters are small dots; the brighter dots represent gate threshold levels for each channel, the darker dots represent the gate release level for each channel, and if enabled in the display settings menu, dots the color of the beatfinder leds which show the gate threshold level for the beatfinder. At the very bottom of the peak meters, an underline will appear when that particular audio channel is being converted into midi data, in either trigger or continuous controller mode.
- **midi data format reference / directload & bpm notelengths [F12]** : when set to display 'midi data format reference', this panel displays a quick reference card covering commonly used midi messages and their data format. For specific information on continuous controllers such as the modulation wheel, refer to the charts at the end of this manual. When set to display 'directload hotkey assignments / BPM notelengths' this display panel indicates any program files associated with the directload keys, as well as the approximate duration (in milliseconds) of various note lengths for the current clock tempo (either from tap tempo, or internal/external clock sources). This can be useful for setting trigger decay times in audio to midi trigger conversion applications.

the menu system

By pressing the Escape key (or any of the function keys from [F1] to [F8]), a menu system 'screen' slides down over the lower display panel, giving access to the menu system which contains all the programmable parameters that define how converter operates. A convenient feature of this menu screen is that it can be retracted (pushed up) and restored (pulled back down) to check performance indicators on the display panel hidden behind the menu screen without losing your place in the menu system hierarchy.

The retractable menu system is navigated by function keys [F1] through [F8], as follows:

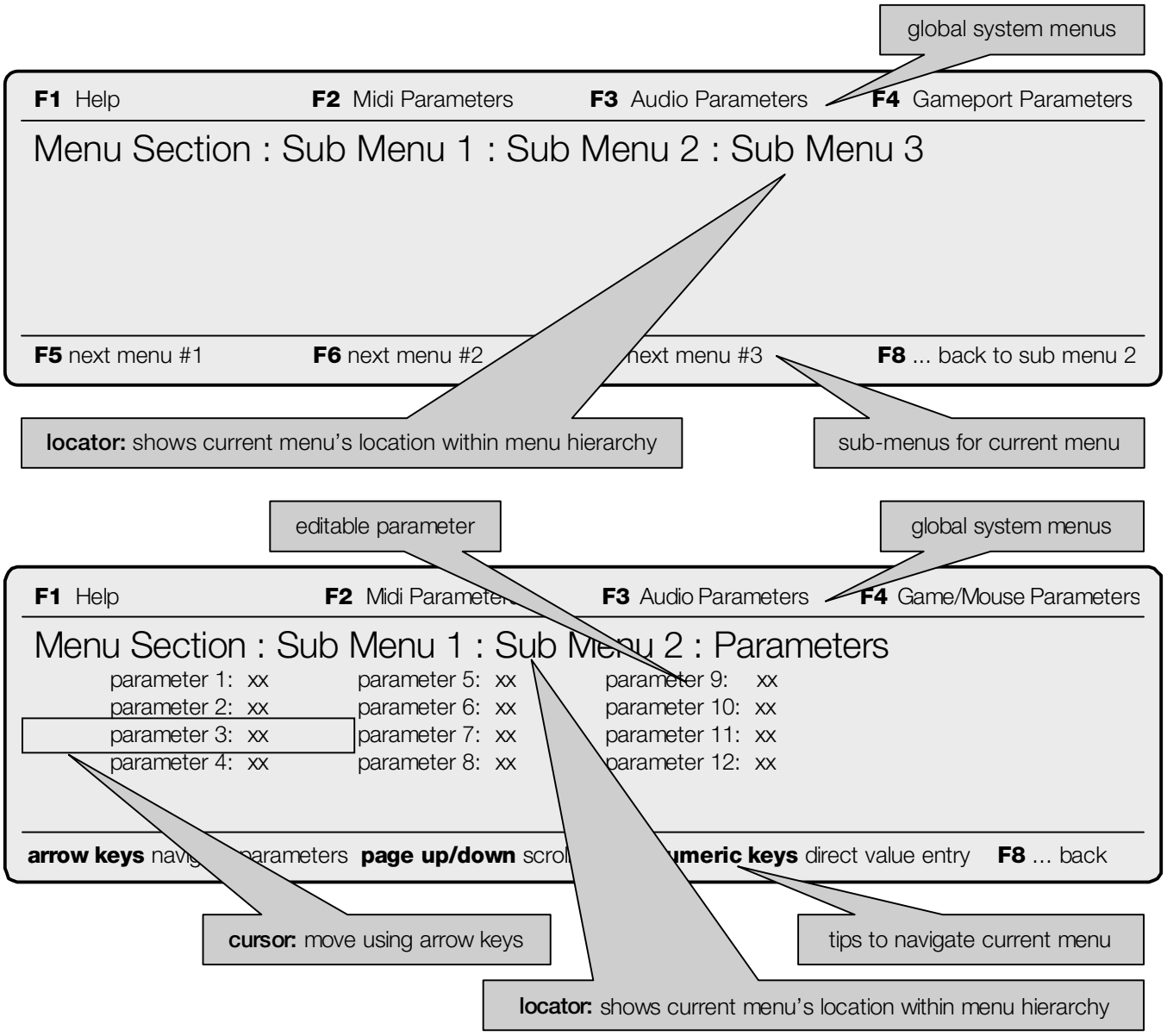
- Function keys [F1] through [F4] serve as direct hot keys to the particular set of menu and parameter pages as displayed on the top bar of the menu screen; these represent discrete sections of the menu system, their assignments never change, and are hardwired. All four of these function keys can be pressed at any time from any location in the menu system, for quick access to another menu section, or even when the menu system itself is retracted and not displayed.
- Function keys [F5] through [F8] are used to navigate through the sub-menu layers within the current menu section as displayed on the bottom bar of the menu screen, and their assignment changes as each menu is entered. When entering a 'parameter page' menu, this bottom field changes to indicate the key presses which can be used to edit the displayed parameters. **However, when in a parameter menu page, the function keys from the previous menu can still be pressed to access neighboring parameter menus directly, without having to back up by pressing [F8] first**. This helps make editing or comparing groups of parameters faster and easier.
- All the above metioned function keys ([F1] through [F8]) can be used to bookmark menu pages you use often for a particular program (.set) file – simply press [Ctrl] and the function key you wish to associate with the page, and use [Shift] and the function key to recall it at any time.

To quickly return to the root menu page at any time, hold down [Alt] and press [Esc]. To jump quickly to specific menus, or from menu section to menu section, there are a variety of hotkeys which are

assigned to specific menus (such as [Alt]-[S] to “save program settings” or [Alt]-[F] for the “audio filter settings” menus); and as stated, function keys [F1] to [F4] can be pressed at any time to return the beginning menus of any of the main menu sections.

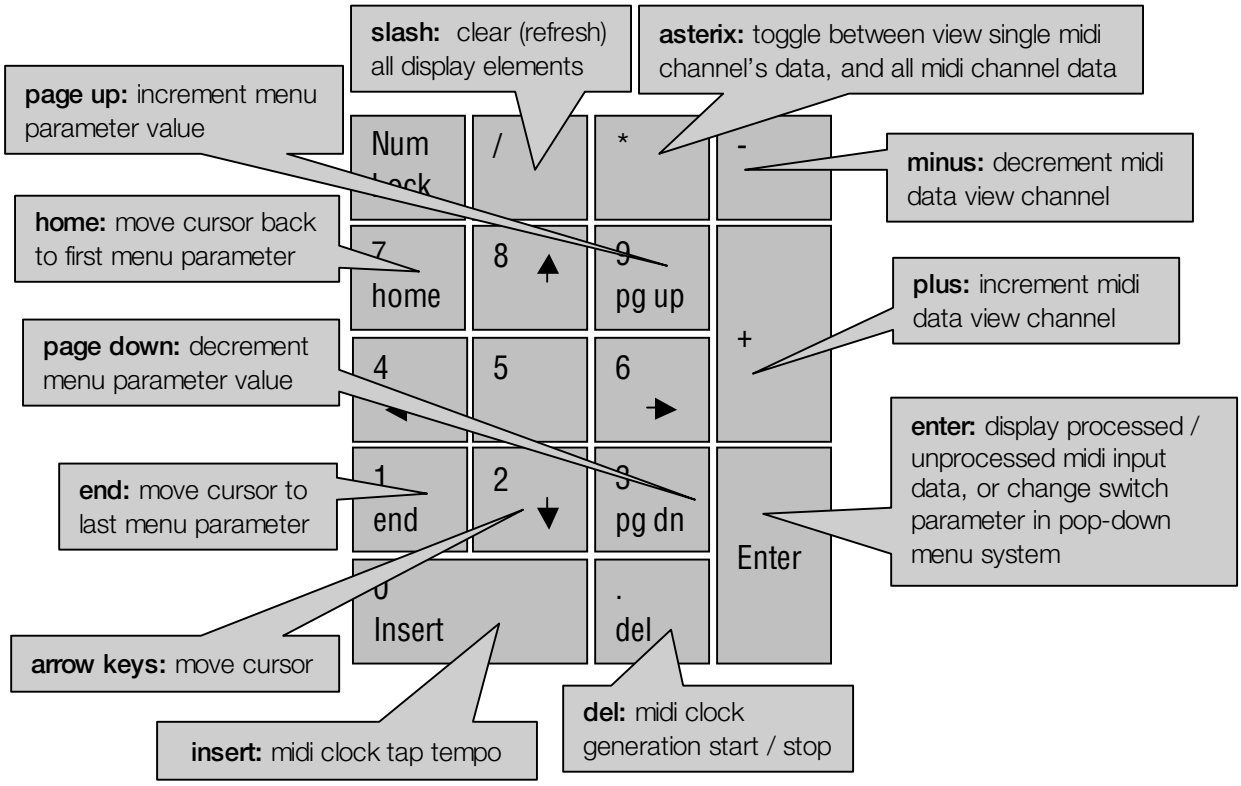
Note that you don’t have to wait for animations to complete before pressing a key or entering data – **try ‘double-clicking’ the function keys**. If animations aren’t desired, they can be disabled from within the “display settings” menu, accessible from the root menu by pressing [Alt]-[Esc], or directly by using the hotkey [Alt]-[D].
While in the menu system, most of the real-time control keys (such as [spacebar] for midi panic or [scroll lock] for system standby) are still available and can be pressed at any time. The only exception to this fact is the [Enter] key, which is used to toggle between displaying raw input data and processed midi data when converter is used to process midi input data. This key is used within the menu system to select files, optionally toggle boolean (on/off or switch) parameters, etc.

Below is a graphical explanation of the retractable menu system which may help to explain the overall working philosophy of the interface.



the numeric keypad

The user interface of converter has been designed around the use of the numeric keypad for quick parameter navigation and data entry. Under one hand rests all the menu cursor navigation controls, midi data view controls, and parameter adjustment controls. Using this direct entry approach can expedite data entry as compared to using a mouse pointer–driven user interface. However, ensure that the 'num lock' key is not depressed if using the numeric keypad in this way. As these keys are duplicated elsewhere on the keyboard, one is certainly not forced to use the numeric keypad in this way.



realtime keyboard controls

Certain keys on the keyboard are assigned functions to control the functionality of converter directly, without having to use the menu system.

midi panic

Every midi system has a use for a midi panic button at some point. The midi panic key is an important tool when using converter, since it is possible to program such things as audio input to control channel volume. In this example, if the audio signal is turned off, channel volume on the target midi device would fall to zero, possibly creating the impression at a later point that the midi device has stopped responding to midi or has crashed. To reset receiving devices connected to converter's midi stream, turn off any stuck notes, and set key controllers to their most useful or default values, press [spacebar]. The only time that [spacebar] will not perform midi panic is when using the menu system and editing a text label parameter.

midi clock generation

There are a group of four keys which are used together to provide quick control of midi clock timing. The [del] or delete key, shown above on the numeric keypad, acts as a sort of 'play' and 'stop' button for the clock generator in converter. The [insert] key next to it serves as the tap tempo button; to generate midi clock data at the tempo of some external music (for example a live drummer or recorded music), tap the tempo of the external music in quarter notes four times. Try to be as precise as possible in the tap timing, especially at faster tempos where minor inaccuracies can have greater impact on the average tempo of the taps, resulting in greater clock drift. Using tap tempo enables midi clock data to be generated at fine resolution fractional tempos unavailable in many hardware devices. To adjust the clock generator's tempo in 1 bpm steps, hold down the [ctrl] key and press the up and down arrow keys. The midi clock generator in converter is automatically shut off as soon as external midi clock messages appear at the midi input port – converter automatically synchronizes itself to any incoming midi clock data (and un-synchronizes itself as soon as the external clock signals disappear).

standby mode

As a complete system "mute" button, or to prevent confusing outboard midi gear with certain midi message streams when programming or experimenting, the [scroll lock] button can be pressed to put converter into standby mode, where no midi data will be transmitted except for clock data if it is being generated.

reset running status

In order to reduce the amount of data required to transmit information between midi devices, the midi standard employs something called 'running status'. What this means is that the statusbyte message (or the first 'byte' of a midi message which tells a device what the message is) is only transmitted once in a stream, unless the message type or midi channel changes. The problem that arises with midi running status is that if a midi device is powered on after that first statusbyte has been transmitted, it won't understand the rest of the stream until it receives the next statusbyte; as a result, it may seem that the device is not responding to midi data or has crashed. When this situation occurs, simply press [R] to 'resend' the statusbyte for the stream; hence reset running status.

quick patch selection

As a way to provide simple and quick patch selection, the [ctrl] and [+] or [-] keys can be pressed. This will increment the patch selection for the current view channel (displayed at the bottom left corner of the screen). If the current view channel is set to ALL, these program change keys will transmit their messages on midi channel 1.

display keyboard controls

The following functions are hardwired to specific keys for quick data display control.

channel-specific data display

Three keys control the data (received or generated) which is displayed: [+], which increments the current display midi channel; [-], which decrements the current display midi channel; and [*] which toggles between displaying all the data on all channels, and just one channel.

clearing and resetting the display

To clear or reset display devices (for instance, the midi message textbox) without affecting the midi data itself, press the [/] button.

turning on or off the midi message textbox

If the scrolling midi message textbox, on the right hand side of the display panel, is preferred to be on or off, pressing the [z] key does the trick.

the upper display panel

Most of the display wedges on the top display portion of the screen should be self-explanatory in terms of what they represent and how they represent it. However, there are a few details of two of the display components which could benefit from a bit of explanation.

the midi message textbox

To the right of the top portion of the screen is a scrolling textbox which, if enabled, shows the generated or converted midi messages. Within the textbox there is two columns of text, one which serves as a text label for the message type, with the particular midi message's relevant numerical value in the right column. Each message is formatted in the following way for greater intelligibility:

- Statusbyte messages (non-"realtime", so in other words voice or channel messages as well as system exclusive) are displayed in bright blue, and the number associated with the message represents the midi channel the message specifies. For example, **CONTROLLER: 8** would represent a controller message on midi channel 8.
- Data bytes following a statusbyte message are displayed in medium-intensity blue, and the number associated with the message represents the value associated with the databyte. If the second databyte represents a note number or a controller number, these numeric values are converted into a more readable text label indicating what note or controller the message represents.
- "Realtime" or system messages (such as clock) are displayed in dark blue, and the number to the right of the message is the specific statusbyte value of the message.

If midi input mode is used and the view processed data parameter in the realtime settings menu is set to off, the midi message textbox displays the incoming midi data values and not the values post-processor.

the system status display wedge

The system status display wedge, located at the top left corner of the screen, displays information about converter's activity in four key ways:

- a processor buffer peak meter, which indicates how much buffer space is being used and is a good measure of the amount of data being converted/generated as well as system loading
- a channel activity led strip, indicating midi messages received or transmitted on one of the 16 midi channels
- five larger leds which indicate conversion activity from the particular input source (in other words, if midi data is being directly generated and outputted from the particular source)
- two banks of red leds which display the continuously moving state of the 8 individual lfos (low frequency oscillators) within converter
- an undefined data display which shows the value of data generated if it is determined the particular midi message is either undefined according to midi specification and might cause confusion to an outboard receiving device (however this does not indicate an error)

keyboard layout and hotkeys quick reference

hotkeys, accessible at any time

[Alt] – [L]	direct hotkey to load program
[Alt] – [S]	direct hotkey to save program
[Alt] – [D]	direct hotkey to display settings menu
[Alt] – [E]	direct hotkey to realtime engine settings menu
[Alt] – [A]	direct hotkey to audio midi data settings
[Alt] – [F]	direct hotkey to audio filter settings
[Alt] – [G]	direct hotkey to gameport midi data settings
[Alt] – [M]	direct hotkey to mouse midi data settings
[Alt] – [N]	to type new descriptive name for current program, displayed on F11 view panels
[Alt] – [X]	quit application
[Alt] – [1] .. [0]	directload hotkeys for fast access to commonly used converter program files
[Ctrl] – [Q]	alternative way to quit application

real time control

[del]	start / stop midi clock generation (automatically disabled by incoming midi clock)
[insert]	tap tempo for midi clock
[ctrl] – [up arrow]	increment current midi clock by 1 bpm
[ctrl] – [down arrow]	decrement current midi clock by 1 bpm
[Scroll Lock]	enter standby mode (halts all midi transmission – essentially a 'pause' button)
[spacebar]	transmit midi panic (note off messages for all 16 channels and controller reset)
[R]	reset midi output running status (re-transmit statusbyte)
[L]	reset lfo waveform to beginning of cycle (ie. downbeat)
[Ctrl] – [+]	increment program change, transmit on current display midi channel
[Ctrl] – [-]	decrement program change, transmit on current display midi channel

display controls

[+]	increment current display midi channel
[-]	decrement current display midi channel
[*]	toggle between view all channels and selected single midi channel
[/]	clear display
[Z]	toggle midi message textbox on/off (unless menu cursor is on a label parameter)
[O]	freeze (or un-freeze) oscilloscopes, when the oscilloscope display panel is selected
[F9]	midi note meter display panel
[F10]	audio oscilloscope display panel
[F11]	hardware and system settings / audio & filter settings display panel
[shift] – [F11]	switch F11 display between hardware & system settings, and audio & filter settings
[F12]	midi data format reference display panel / directload hotkey assignment panel
[shift] – [F12]	switch F12 display between midi format reference & directload hotkey assigns.

note: When the [Alt] or [Ctrl] keys are referred to in a key combination such as [Alt] – [Esc], it is meant that the [Alt] key should be held down while the [Esc] key is pressed and released.

accessible within the menu system display

[Esc]	pop-down / pop-up menu system display (remembers current menu page)
[Alt] – [Esc]	direct hotkey to the root menu
arrow keys	move cursor
number keys	direct value entry
alphabetic keys	direct label entry
[home]	move cursor to first parameter in menu page
[end]	move cursor to last parameter in menu page
[pg up]	increment parameter value
[pg dn]	decrement parameter value
[Enter]	select program, operation, or file; toggle parameter on or off
[F1] – [F4]	select global menu section
[F5] – [F8]	navigate current menu system
[Shift] – [F1] .. [F8]	user assignable bookmark hotkeys, for direct access to frequently used parameters
[Ctrl] – [F1] .. [F8]	assign the particular bookmark hotkey to the menu page currently displayed
[Tab]	audition key (only in audio midi data settings menu) to hear note programming

accessible within midi program & operation selection and settings menu pages

[Alt] – [C]	copy program / operation
[Alt] – [P]	paste copied program / operation to current slot

accessible when the menu system display is retracted

[Enter]	toggle between view processed midi data and raw input data display modes (applies to midi input processor, not audio/gameport/mouse input conversion)

midi clock generator

In addition to being able to synchronize its lfo generators to incoming midi clock, converter also features its own internal midi clock generator. This clock generator was designed to be fast and direct to control, making it optimal for use in a live performance application, or in jam / improvisational sessions in a studio, where acoustic (human-played) rhythm sections define the tempo of a given performance. To enable this type of application, converter provides tap tempo functionality (insert key, or 0 on the numeric keypad) and a dedicated start / stop key (del key) for its midi generator.

In order for converter to clock as accurately to the given musical situation at hand, it is important to tap the desired tempo as accurately as possible. Even though not displayed on the screen, converter uses high internal precision to enable it to generate fractional tempos. The tempo displayed on the screen is the fractional tempo rounded to its nearest bpm.

Keep in mind that almost any midi clock source will eventually drift over time, even two separate clock sources which are set to the same bpm (which is why a single midi clock source is always used in recording applications), and as a result it is important to make playing decisions based on this fact – for example, if sync'ing a synthesizer's arpeggiator to converter's clock in a band situation, don't set the arpeggiator to its indefinite 'hold pattern' position; use a key triggered or one shot mode instead.

When incoming midi clock is received by converter, the midi clock generator is automatically halted, converter auto-syncs to the incoming clock messages and the tempo becomes that of the external clock source.

While the new, overhauled clock generator in version 1.4 provides a far more stabilized clock source than previous versions of converter, clock accuracy can be improved (if necessary) by using converter in one of its midi input only modes (without audio input), due to the interrupt frequency and priority of the audio input code versus the interrupt priority of converter's midi clock generator. However, the difference in accuracy is slight, and would likely only be an issue if converter's midi clock is being used in certain recording situations demanding a large degree of precision. Certainly for live performance applications this should not be an issue.

Metronome 'click track' functionality is embedded in converter, providing a useful aural cue source for human performers in a live situation where they must follow the tempo of an electronic source. The click track is generated either by the internal clock in converter, or auto-synchronized to external clock received from another source such as a sequencer. Refer to the section in this manual on Auxiliary Realtime Settings for information on configuring this feature.

directload keys

By using the directload keys, commonly used programs (.set files) can be quickly loaded for a given application. In this way, converter can be instantly re-configured at the touch of a button, saving time otherwise spent looking for a particular program file. For example, templates for individual pieces of vintage midi gear, audio loop source material, particular songs in a live performance, or controller configurations can be recalled as soon as the sources are patched up to the interface connections converter is using on the computer.

The directload keys are assigned to specific files by entering the “load program” menu page where files are listed, moving the menu cursor over the desired program file, and pressing [Alt] in combination with the directload key (number keys 0 to 9) to be associated with that program file. When not in the “load program” menu, in other words at any other time in converter, the assigned file is instantly loaded by using the same combination of the [Alt] key plus the assigned number key.

The directload key assignments are automatically saved in converter’s configuration file c.cfg upon exiting converter, and remain the same until their assignments are specifically changed at some later point. In other words, loading a new program (.set) file will not affect the directload key assignments.

To check which files are assigned to individual directload keys, the directload viewpanel can be viewed by pressing the [F12] (or [shift]+[F12] if the midi data reference panel is displayed by default) and is viewable when the pop-down menu system is not in view.

To help ensure that work is not accidentally lost, all current settings are saved to the file ‘lastset.set’ before loading the new file associated with the directload key. If a directload key is accidentally pressed, simply load ‘lastset.set’ from the load file menu to recall the last (unsaved) settings.

The default settings assigned to the directload keys are the presets which accompany converter; these assignments can obviously be changed, if desired.

bookmark hotkeys

There are 8 keys which can be used for quick direct access to menu pages you use most frequently for a particular program (.set file). These 'bookmarks' can be recalled at any time by holding down [shift] and pressing a function key from F1 to F8. To assign the hotkeys, simply hold down the [ctrl] key and press the function key (F1 to F8) you wish to assign the menu page to.

To see which hotkeys are assigned to which menu pages, there is a dedicated view panel provided. If the retractable menu is down, press [esc] to push it up, and press [F12]. Then, hold down the [shift] key and press [F12] until the "Menu Page Bookmark Hotkey Assignments" view panel is displayed, if it isn't already visible.

Quite simple, really…

menu pages and parameter descriptions

The following pages of this manual are segmented in five sections, discussing the global settings for converter, followed by its audio, midi, gameport, and mouse input functionality. Each section devotes a page (or two) to each individual page in the menu system, and outlines the parameters found on it and describe their usage. At the top of each page, the menu page's title, location in the menu system hierarchy, as well as the key presses to access it from the root menu, are listed.

The menu system employs animations to the text. Keys can be pressed at any time – there is no need to wait for any animations to complete before the next key is pressed. To speed up, slow down, or turn off the animations, refer to the page discussing the **display settings** menu page.

disk i/o

Root : disk i/o

[Alt+Escape – F5]

Program files store all the midi processor programs, audio, gameport and mouse settings, input source configurations, several of the global system settings as well as the custom viewpanel settings. These program files are stored in the same directory as the converter software; the program files are identified by their '.set' extension. A feature of converter is that the current program is automatically saved at exit into the file `default.set`, and immediately re-loaded upon re-launching the program – so if the same system configuration is always used, converter is always ready to operate immediately.

load program

Allows the selection of a program file (.set) to be loaded for operation. There is no support for directory hierarchies in the load function – all 'programs' have to be stored in the same directory as converter itself (not great but that feature is (hopefully) most likely not necessary)

This menu can be directly accessed at any time by pressing [alt]-[L].

save program

Allows you to save your programming work - useful feature! Just type in the filename (the filename input box doesn't allow you to use your own extension - .set is automatically added to the filename) and press enter.

This menu can be directly accessed at any time by pressing [alt]-[S].

turn midi logfile on/off

Enables or disables the continuous storage of all input or generated midi data messages to a file called `logfile.txt` in the same directory as converter. This text file can be loaded in any text file viewer (such as Edit under DOS or Notepad under Windows), and its contents are formatted in the same way as in the scrolling message textbox on screen. This feature is best left off, and is merely provided for those using converter as a means to debug midi message streams generated by other midi devices, or some other type of technical work.

Note that this feature is unavailable when converter is being run from a floppy disk due to the inherent lack of speed of the floppy disk drive – converter must be run from a hard disk for this function to be visible.

core realtime engine settings

root : realtime settings : core realtime settings

[Alt+Escape – F6 – F5]

midi input processor
Enables or disables (bypasses) the midi processor, which is used only for incoming midi data (coming from the 'midi in' connector on your midi interface). This parameter can be left at the 'on' position at all times.

view processed data
Selects whether the display panels show data after or before passing through the midi input processor. By setting this parameter off, the raw input (unprocessed) midi data is displayed.

midi clock generator
Enables or disables the midi clock generation functionality of converter. Note that clock generation can also be initiated by pressing the [del] key. The difference between the two methods of enabling converter's midi clock generator is that the [del] key generates midi start and stop messages when it is used to start and stop midi clock; when this parameter is used instead, midi clock is started or stopped without actual midi start or stop messages being generated and transmitted (useful for situations where it's preferable not to trigger external sequencers or arpeggiators).

default clock tempo
defines the tempo of the midi clock generator, in BPM. Valid values are between 20 and 250.

audio input
Enables or disables all audio input midi conversion (including the audio filter channels). Note that audio signals will still appear in the oscilloscope and audio / hardware settings display panels, and incoming audio streams can still be used as modulation sources for the midi input processor. By disabling this parameter, all audio channels no longer generate their own midi streams or trigger data.

audio main bandpass
Enables or disables the use of the bandpass filters associated with the main audio input channels.

audio left filter / audio right filter
Enables or disables the left or right filter channels.

Note: main audio input must be enabled in order to use the filter channels. If midi data conversion from the main/bandpass channels is not desired, simply set their conversion mode to 'no conversion' in the **audio midi data settings** menu.

gameport input
Enables or disables input from the PC's gameport. By disabling this input, system speed is increased, due to the computing-cycle-wasting requirements of the analog gameport. Unless gameport input is being used, it is recommended that gameport input be disabled. Also note that gameport input is automatically disabled when converter is run in audio mode with a buffer size less than 72 bytes – otherwise audio interrupts occur so frequently that converter cannot accurately acquire proper axis readings.

mouse input
Enables or disables input from the PC's mouseport. Mouse to midi conversion doesn't create any sort of significant computing load; however, this enables mouse input to be disabled if it is preferable to avoid any undesired midi messages transmitted by accidentally moving the mouse, etc.

audio sum to mono
Determines whether the stereo input signal is summed (merged) together to provide the same mono signal to both left and right audio channels. Useful for 6-band audio filter bank applications. Keep in mind that if there is much of a phase difference between the left and right channels, amplitude of low frequencies (or possibly higher frequencies as well) may be reduced.

lfo generators
Enables or disables the bank of 8 lfos. By disabling the generators, all calculations and midi data generated by the lfos is turned off, and will disable its use as a modulation source (if used) for the midi input processor. Oh, and the lights will stop glowing too. ☺

This parameter should normally be left at ON – use the settings in the LFO Generators menu section to control whether or not the lfo generators transmit midi data or not.

auxiliary realtime engine settings

midi clock click
Enables or disables the midi metronome 'click' generator. This function generates a metronome reference using a specific midi channel and note number, and can be used as a tempo cue for live performers.

click midi channel
Determines which midi channel is used to transmit the note on / note off messages for the metronome clicks. Default is channel 10.

click midi note
Determines which note is used for the midi metronome click. Default is 76 (high wood block in the general midi drum map specification).

click velocity
Determines the velocity of the clock clicks. On each downbeat (quarter note #1), the click is transmitted with a velocity of 127; for the other three beats, the clicks are transmitted using this parameter's velocity level. If an unaccentuated downbeat is preferred (if using a time signature other than 4/4), simply set this parameter to 127 for all metronome clicks to have the same velocity level.

display settings

root : display settings

[Alt+Escape – F7]

The default settings for the following parameters will usually be the best settings for most situations on most computers. However for reasons of personal taste or limited computing resources, converter can be customized easily on the fly with the following options.

update screen

Selects one of three screen update settings: **on** (everything is refreshed), **minimal** (only the system activity panel is updated), and **off** (none of the graphical elements are updated in real-time). By selecting off or minimal, the computer completely focuses exclusively on performing midi processing or conversion operations. In general, converter's graphics code is performed at a low priority respective to the time critical processing and/or conversion, so typically this parameter can be left set to "on". However if the only machine you have to use is an old 386DX system or you otherwise feel the computer is severely bogged down and could perform better with a lighter load, and you don't need to see what the computer's doing, this option may improve system response times.

vertical retrace

Enables or disables screen redraw synchronized to the monitor's vertical retrace interrupt. Enabling this parameter will result in less flicker from screen updates in certain situations; however it will slow down the speed of the graphics routines when there is a high processing load, and may negatively affect midi performance. An alternative, better way to reduce flicker is to set the oscilloscope redraw parameter to 'intelligent' (see below).

animations

Enables or disables user interface animations.

animation speed

Sets the speed at which the animations are performed.

midi message textbox

Enables or disables the scrolling midi message textbox at the right of the display panel. The updating of the text messagebox is only performed when free computing cycles are available, therefore having it enabled shouldn't affect performance; however it is sometimes beneficial to disable this as it does add an additional load to the graphics system. In many applications (such as continuous controller audio to midi conversion) the stream of generated midi data will scroll by too quickly to read, and so computing cycles are wasted on a flurry of illegible text.

oscilloscope redraw

Selects the mode in which the oscilloscopes are redrawn. In **intelligent** mode, the oscilloscopes are updated only when a midi message is generated from the current input(s). In **continuous** mode, the oscilloscopes are updated continuously, even when the entire system is idle and there is no input signal. Intelligent mode provides a way to reduce unnecessary flicker in the oscilloscope displays, however continuous mode is more suitable for situations such as trigger mode where little midi data is actually generated from the audio inputs. Normally, leave this set to continuous.

translate numerics
Determines whether or not converter displays midi statusbytes, note numbers, and controller numbers in their numerical format or as a statusbyte label with channel number or note letter with octave or controller label for the various parameters within the menu system which directly reference midi bytes. By enabling this option, for example, parameters will display a statusbyte as “poly aft. [ch. 12]” for a polyphonic aftertouch message on midi channel 12 rather than that same statusbyte’s actual numerical representation of 171, as well as displaying “C octave 4” rather than that note number’s numerical value of 60.

Possible settings are OFF, ON, and ON+GM DRUMS.

When set to ON+GM DRUMS, note values referencing midi channel 10 are displayed as the equivalent percussion instrument associated with the general midi standard, to make quick programming of drum maps / triggering. If not using general midi devices, or it is preferable to treat midi channel 10 as any other midi channel, simply set this parameter to “ON”.

It is recommended this parameter be left at one of the ON settings, unless you are used to (and enjoy) working with midi data in decimal format. ;-)

aluminum look
Enables or disables the “brushed aluminum” look in converter. Leave it on, it looks better :-)

beatfinder
Enables or disables converter’s ‘beatfinder’. Beatfinder is a simple feature in converter which assists in the setting of gate threshold parameters, and tweaking rhythmic / trigger-based audio to midi conversion in general. By enabling beatfinder, a bank of ‘leds’ above the audio peak meters in the audio display panels [F11] indicate when that particular audio channel’s signal level has crossed the defined gate threshold. Unlike other visual guides in converter such as the peak meters, the beatfinder led’s are response-guaranteed, in that they will always indicate the threshold crossing, even if it occurs for just half a millisecond. This provides a clear visual trigger indicator when programming, or for quickly verifying levels when in a live or studio situation.

threshold to use
Determines if beatfinder should use its own gate threshold, which is the same across all 6 audio channels and useful in initial level setting, or each channel’s individual gate threshold, to aid in specific tweaking of trigger thresholds.

beatfinder threshold
Determines the level of the beatfinder’s own gate threshold.

color scheme
Allows the selection of one of four different color schemes for converter. How pretty.

customize user panel

root : customize user panel

[Alt+Escape – F8]

panel meter (A – E)

Selects the midi controller message type to monitor for the particular meter in the ‘custom user panel’ display wedge. This provides a visual monitor for controller types not assigned to existing hardwired display elements on the upper portion of the screen.

Audio to midi conversion in converter is performed on the basis of amplitude, in one of two modes: a continuous time-varying conversion mode for continuous controller applications, and a trigger/gate conversion mode useful for using an audio source to trigger midi note on / note off messages. **Note that only the line-input is used on all the soundcards – the mic input is not supported.** On the SoundBlaster 16, SoundBlaster Pro and compatibles (as well as the Gravis Ultrasound), converter automatically sets the level for the line input to its maximum position so that an external mixer program is not needed.

Each audio channel generates a stream of single value bytes, based on the amplitude of the incoming audio, which can be inserted into any data byte in a particular midi message type. The associated midi message data which encapsulates the converted value bytes is defined in the audio parameters menus.

As an example, the following steps would program the audio conversion engine to generate a continuous stream of modulation wheel data (continuous controller) from the left (non-filter) input channel. This is assuming that the main audio input is enabled and converter is operating in audio input mode.

Step 1: program the midi conversion settings

- go to the audio parameters menu section (F3), press [F7] for the midi conversion settings menu, then [F5] for midi data settings, and finally [F6] for the main audio channels parameters.
- type 2 for the left conversion byte position parameter, 176 for the left statusbyte (or use the page up / page down keys to scroll until the parameter reads “cntrl [ch. 1]”), then type 1 for the left 1st databyte value (or scroll until the parameter reads “mod wheel”), and leave the left 2nd databyte value at zero. This tells converter that the converted value byte should be used for the 2nd data byte in the midi message, and that the type of midi message to generate is a controller message on midi channel 1 (by using 176 as the status byte). By setting the 1st data byte in the message to a 1, we have specified the controller message type to be modulation wheel.
- move the cursor to the left convert mode parameter and choose continuous controller mode (“cont. ctrl”), then select “no conversion” for the right convert mode parameter.

Step 2: specify the midi data reduction settings (optional)

Since continuous-controller type of audio conversion can generate a tremendous number of midi messages per second, especially if more than one audio channel is being used, it can be useful to program the data thinning algorithms to intelligently reduce the density (messages per second) of the generated midi stream.

- press [F8] twice to back up in the menu hierarchy, then press [F6] for data reduction settings.
- there are two parameters for each of the six audio channels; the parameters for the main audio channels (non-filter channels) are the first four on the left of the menu screen. Try a value of 5 for the left continuous controller reject parameter (labeled “left cc reject” on the menu), and a value of 6 for the left channel reject threshold parameter (labeled “left reject thresh.”). These settings mean that out of every 5 audio byte conversions, only 1 is transmitted, unless there is a sudden change in the audio amplitude that exceeds the threshold (which we set to 6). If the latter happens, a midi message is transmitted instantly (data rejection does not occur) in order to preserve the musically important transients of the audio, while reducing the amount of unnecessary midi data.

That’s all there is to it. If for some reason no data is being generated (and yet you can see audio input in the oscilloscope or system / audio info display panels), check to make sure the input gates are not interfering by setting the left threshold parameter to 0 in the main audio gates settings menu. Additionally, you may want to experiment with the arithmetic processing that can be applied to the channel allowing such processing as amplitude compression or expansion (using a combination of division and addition or multiplication and subtraction operands respectively).

The following is an example of how to program converter in trigger mode, so that impulses on the left main audio channel will trigger a note with velocity on midi channel 1.

Step 1: program the midi conversion settings

- go to the audio parameters menu section (F3), press [F7] for the midi conversion settings menu, then [F5] for midi data settings, and then [F6] for the main audio channels parameters.
- type 2 for the left conversion byte position parameter, 144 for the left statusbyte (or use the page up / page down keys to scroll until the parameter reads "note on [ch. 1]"), then type 60 for the left 1st databyte value, and leave the left 2nd databyte value at zero. This tells converter that the left main audio channel converted value byte should be used for the 2nd data byte in the midi message (representing note velocity in this case), and that the type of midi message to generate is a note on message on midi channel 1 (by using 144 as the status byte). By setting the 1st data byte in the message to a 60, we have specified the note number (middle C in this case).
- move the cursor to the left convert mode parameter and choose trigger mode ("trigger"), then select "no conversion" for the right convert mode parameter.

Step 2: program the audio input settings

- press [F3] to quickly return to the beginning of the audio parameters menus, then [F5] for the audio gate settings menu, followed by [F6] for the main audio channel gate settings menu. The three parameters we need to check and / or modify are the "left trigger level", "left release level", and "left trig decay time".
- the left **trigger level** parameter controls the input gate, in that the audio signal's amplitude must exceed this level before a midi message is generated from it. This can be set to any value from 0 to 127; however with a value of 127, audio will never be converted because 127 is the max level. With a value of 0, audio at all amplitude levels is converted (the gate is effectively disabled). This feature is useful to separate wanted signals from unwanted signals (ie., separate signals from background noise, or eliminating accidental triggering from other instruments bleeding into the microphone such as a hi-hat bleeding into a snare drum mic). This can be set at 0 for now, and adjusted if the source being tracked requires a different setting.
- the left **release level** parameter determines the level below which the audio amplitude must fall before a zero-value midi message is generated (in this case, note on velocity zero which is equivalent to a note off message). This can be any value from 0 to 127; however if set to zero, a zero-value midi message will never be generated. Since the notes will be stuck on without a corresponding note off – type message, converter automatically sets this parameter to 1 when the audio channel is set to trigger mode. Raise this value higher if the particular audio source signal you are using for the channel tends to never drop to zero.
- The left **trigger decay time** parameter determines the minimum time window (in milliseconds) between a trigger value midi message (in this case note on) and its subsequent zero-value midi message. This parameter is useful to avoid rapid re-triggering of notes (mis-tracking of the audio signal), especially if using a very small audio buffer size where the actual rise and fall of lower frequency waveforms may be traced. Note however that this parameter can limit the minimum note length that converter will track at a given tempo / bpm for the particular channel, and provide less useful results. For example, the higher the number used, the fastest note length converter will accurately track will move down from 64th notes, down to 32nd notes, down to 16th notes, etc. Try a setting around 80 to start, and adjust as necessary.

And that's it – converter is programmed to perform triggering based on audio taken from the left audio input. This same methodology can be applied to the right channel, or any of the other audio channels. Note that often it is beneficial to use a filter on the audio channel for conversion work rather than the full bandwidth input audio signal, as the filters provide the ability to further separate wanted from unwanted trigger audio impulse signals.

Additionally, you may want to experiment with the arithmetic processing that can be applied to the trigger channel, through which one can create customized velocity response curves and the like. An interesting way to shape the sound of the triggered midi note is to assign a different audio channel

to continuous controller conversion, and program the channel to generate channel volume messages for the same midi channel as the trigger note. In this way, the volume and shape of the triggered note follows the incoming audio signal, and can integrate itself quite musically.

Note that setting the levels of the incoming audio is very important, and has a great affect on the speed and accuracy of trigger timing and predictability. If the occasional trigger seems late, it most likely is the result of the audio input levels being too low so that only the very tip of the audio impulse is captured. Try raising the volume of the audio going into your soundcard.

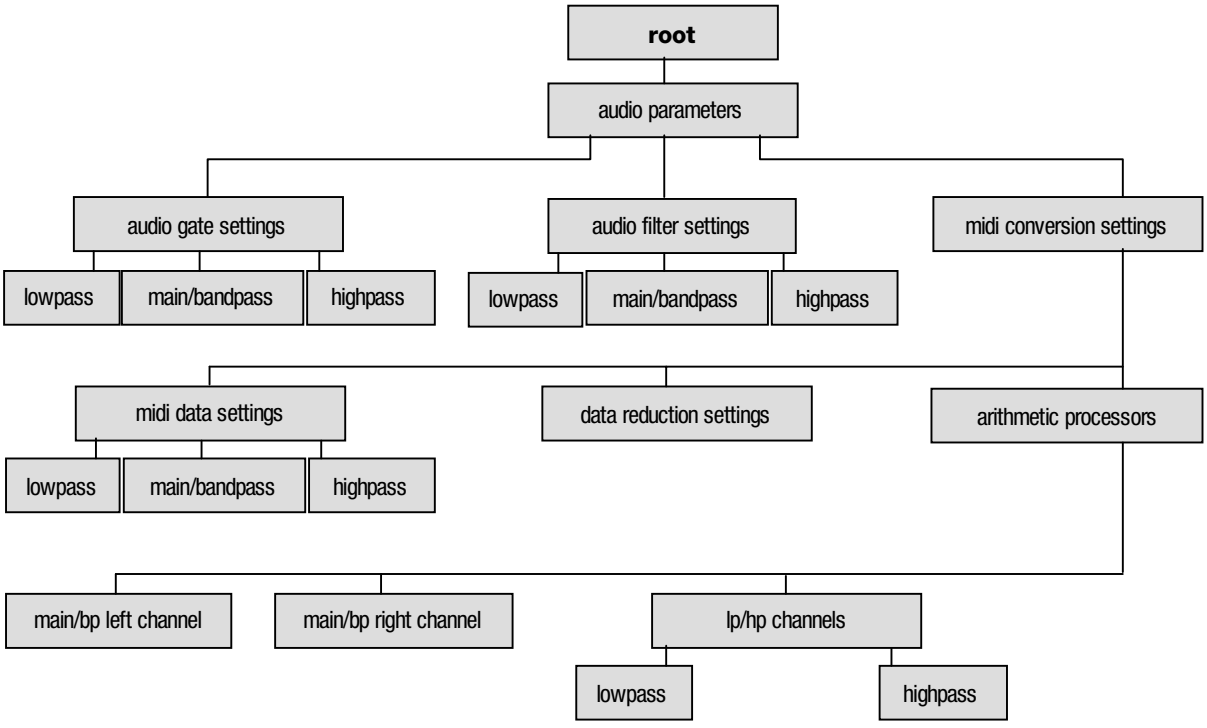
Since the input audio is never heard once it reaches converter, it is not necessary to avoid clipping the signal – in fact, it can be useful to overly increase the input level of the audio in certain applications. In trigger mode, it is usually better to set the signal level so as to avoid constant input overload: the signal level should be set to ensure the audio peaks rise and fall within the trigger gate thresholds, to avoid mis-firing.

Obviously, audio sources vary significantly in terms of bandwidth and amplitude, so the key is to experiment with different source material and try different ways to program the conversion process. Take advantage of the gates on each audio channel to help separate signal from noise or other unwanted sounds. You may find that using an external equalizer to further filter out unnecessary frequency sections of an audio source or boost specific frequency areas will greatly improve the possibility for signal separation between the low and high pass filter channels.

Keep in mind that trigger mode conversion works best and most reliably with a non-mixed audio source – for example, a single track consisting of a snare drum rather than a channel from a stereo mixdown with many layers and instruments. While converter has been designed in such a way that it is possible to 'sample' rhythmic information from complex, layered audio source material, trigger mode was intended more for applications such as triggering drum samples from a multi-tracked acoustic drum performance or a live mic'd drumkit than from a dynamically-compressed mixdown.

Note: Even though selectable as a message type, audio, gameport, and mouse input cannot be converted to system exclusive (sysex), mtc/smpte ¼ frame, or certain undefined system common/realtime message types.

The following is a diagram of the menu hierarchy of the audio section of the menu system.



audio gate settings

Depending on the audio conversion mode selected for a particular channel, one of two different sets of parameters will be visible for that channel in the audio input settings menu.

trigger decay time

When using a smaller input buffer for audio, or if converting a low-frequency waveform, the system may analyze the audio input quickly enough to generate a trigger pulse for each cycle of the input waveform. To avoid this rapid re-triggering, a minimum time constant (in milliseconds) can be established within which a subsequent re-analysis of the audio input will not be performed. This parameter can be ignored if trigger mode conversion is not being performed for the particular channel. If a certain note length is desired as a forced maximum tracking speed, the 'bpm notelengths' display ([F12] or [shift]-[F12]) in combination with converter's tap tempo feature can provide a rough idea of what value to use for this parameter.

WHEN CHANNEL IS SET TO CONTINUOUS CONTROLLER CONVERSION MODE:

threshold

This parameter determines the gate cutoff level for the respective channel (left / right), for which the audio signal's amplitude must cross before being converted, and below which audio will not be converted to midi information. As an example, this is useful for converting audio signals which have a higher residual noise component than usual. Possible values are in the range of 0 – 127.

resting value

Typically, when the audio falls below the gate cutoff value (threshold) for a particular channel, that channel's conversion value equals 0. The resting value parameter allows the zero-audio level to be any value from 0 – 127, useful for continuous controller applications where the controller should rest at a midpoint of 63 (such as the pitch bend controller), or default to a max value at 127.

WHEN CHANNEL IS SET TO TRIGGER CONVERSION MODE:

trigger level

This parameter specifies what amplitude the audio must cross (above) before a trigger event will be generated. Essentially the same as the threshold parameter in continuous controller conversion mode, the range of possible values are between 0 and 127.

release level

This parameter specifies the amplitude level below which the audio must fall before a release event (ie. note on velocity zero) will be generated. Possible values are between 0 and 127, however setting this parameter to zero will result in a release event never being generated (the audio amplitude cannot fall lower than zero!). This parameter is automatically set to 1 when the particular audio channel has been set to perform audio to midi conversion in trigger mode.

audio filter settings

filter type
Selects the type of filter to be used for the particular audio channel. Possible choices are shelving, bandpass, and peaking EQ for lowpass and highpass channels. For the main/bandpass channels, a choice of either bandpass or peaking EQ is available.

frequency
Selects the cutoff frequency in Hz for the filter for the particular audio input channel. Possible values are between 20Hz and 5000Hz for lowpass channel filters, 20Hz and 10512Hz for bandpass channel filters, and 1000Hz to 10512Hz for highpass channel filters.

Q
Selects the Q (bandwidth or sharpness of the peak) of the filter. Possible values are between 1 and 15.

gain
When using a peaking EQ filter, this is the amount of gain applied to the center frequency, in dB. For other filter types, it serves as an overall gain for the channel's signal (post filtering).

audio midi data settings

When programming an audio channel to generate a midi note on message, the particular note can be 'auditioned' by pressing the [Tab] key while the menu cursor is hovering over one of the channel's midi data parameters. This is particularly useful when programming drum triggering.

converted byte position

Selects which databyte of the chosen midi message the converted value is placed in. For example, to use a channel of audio input to control the pitch wheel, the generated message would consist of one statusbyte and 2 databytes as follows: the statusbyte (indicating a pitch wheel message on midi channel X), followed by the first databyte representing the LSB component of the pitch wheel message (unused in this case since we can only provide an 8-bit value, therefore should be set to a value of zero), followed by the second databyte representing the MSB value for pitch wheel. Hence you would type 2 or scroll the parameter until it says "databyte #2" (to indicate that the value converted from the audio input signal should be placed in the 2nd databyte of the midi message). Since the statusbyte is not considered a data byte, this parameter should be read in terms of databyte number, not byte number in the cumulative message packet (i.e. 2 instead of 3). Possible databyte position values are either 1 or 2; if set to zero (or "unused in msg"), the converted byte is not added to the midi message to be transmitted (in this way, a fixed message will be generated when the audio is above the gate threshold).

Note that not all midi messages use both, or any, data bytes; channel aftertouch, program change, and song select only use the first databyte (for aftertouch amount, program change number and song number respectively), while system realtime messages such as midi clock simply use the statusbyte. As converter automatically enforces these midi definitions, setting this byte position parameter to 2 when the message type only supports one data byte results in the converted value not being a part of the generated midi message.

statusbyte

Determines the midi message type (and channel) to use for the audio conversion value. Possible values are between 128 and 255.

1st databyte value

Sets a fixed value for the first databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 1st databyte position.

2nd databyte value

Sets a fixed value for the second databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 2nd databyte position.

convert mode

Selects the mode in which to perform audio to midi conversion for the particular channel. In continuous controller mode, audio amplitude is tracked to provide time varying continuous controller values. In trigger mode, the audio is monitored to generate a midi message when the audio's amplitude exceeds the gate's threshold, with a respective counter-message generated when the audio falls back below the gate's threshold. Typically, trigger mode conversion is used to generate note on / note off messages from such sources as percussion sounds, etc.

audio midi data reduction settings

The parameters on this particular menu page only impact continuous controller audio to midi conversion; trigger mode audio to midi conversion does not employ (or require) data reduction.

continuous controller reject (cc reject)

Selects the ratio of converted values to transmitted midi messages. For instance, with a value of 0, every analysis value is converted into a midi message and transmitted (most often resulting in a very dense stream of midi data, especially if all six channels of audio are being converted with the same stream density). With a value of 4, only every fourth value is converted into a midi message and transmitted. Possible values are 0 to 99, with a value of zero disabling data reduction.

reject threshold

This parameter controls the amplitude change upon which data rejection is temporarily abandoned in order to allow sudden fast peaks to be accurately converted and transmitted immediately. This allows for data to be thinned when there is little change in the amplitude of the input waveform, yet gives priority to sudden and musically important shifts in amplitude. Possible values fall between 0 and 127; with a value of zero or one, any change in amplitude is given priority which effectively disables data reduction.

Note that the arithmetic operations are performed sequentially in the order they appear in the transform list, from first to last – 'order of operations' is not followed. This allows for the input data to be processed in a more flexible manner.

transform operand (X-form)

Selects the type of data transform to be performed on the value converted from the audio input signal. Possibilities are: no xform, add, subtract, multiply, divide, and invert.

transform value

Selects the numeric value for the transform. For instance, if the chosen transform is divide, and the transform value is 3, all converted values will be divided by three. Possible values are between 0 and 127 (protection is implemented against division by zero).

of transforms

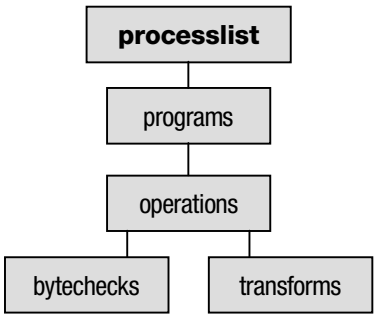
Determines how many transforms are active, counting from the first transform. This makes it easy to bypass certain transforms and only activate the first 1, or 2, or 3 transforms. If set to zero, none of the transforms will be performed on the converted data.

midi input processor

The architecture of the midi processor is structured into 3 levels: the **process list**, which is the list of programs performed (in sequential order) by the processor; a **program**, which represents a group of **operations;** and the **operations** (also performed in sequential order) which directly perform the midi data modification through a number of **bytechecks** (or conditional verifiers) and **transforms**. Upon receiving a complete midi message, the midi processor cycles through the process list to see if the midi message type should be modified by any of the operations in any of the programs.

The specifics of this architecture are as follows:

- the process list can contain up to 12 programs
- each program can contain up to 12 operations
- each operation consists of 1 statusbyte comparator (found in the operation parameters menu), 3 data byte checks, and 4 transforms which are performed if the current midi message conforms to any of the data byte checks used.
- each operation functions on one specific midi message type (statusbyte value)
- more than one program or operation can modify a given midi message (useful if the desired modification requires more than 4 transforms to complete, such as sysex)



To illustrate how to apply this system to a practical task, let's consider the design of a program to perform a keyboard split function at middle C for incoming midi data on channel 1 that will send all note messages below middle C out on channel 4.

IMPORTANT!
Whenever creating an operation or program which works with **note on** messages, also ensure that **note on** messages with a velocity of 0 are dealt with properly (for example, passed over by the midi processor via the use of the 'byte-check' operations), in the same way as corresponding **note off** messages, in order to avoid stuck notes.

There are two ways in which keyboards transmit note information; one involves the use of both note on and note off messages, and the other involves only note on messages which are considered note offs if their velocities are equal to zero. The program should be designed to support both methods of note off message transmission in order to function correctly with all keyboard controllers; we will design our program to actually convert note on messages with a velocity of zero to actual note off messages to illustrate one of the ways in which converter can be used to change the formatting of the midi stream.

Here's how to implement our keyboard split program:

- select an empty program slot in the process list (F2), in our case it should be slot #1 (if there are other programs already on the processor and you don't know what they are, you can start with a 'blank slate' by pressing [Alt]-[L] and loading EMPTY.SET)
- go into the program settings menu (F5), label the program as "keyboard split", and set the **number of operations** parameter to 3. **It is a good idea to always set this parameter to the exact number of operations actually used in the program** in order to avoid possibilities for programming errors, even when the other operations are left to their default values and have been untouched.

Now we have our program. The next step is building the three operations we need for this task:

- enter the operations list for this program (F6) and select the first slot.
- go to the operation parameters menu (F5), label the operation as "note on ch. 1", set the **message type** to 144 (note on channel 1), set the **# of byte checks** to 2 and the **# of transforms** to 1.

- go to the bytecheck parameters menu for this operation (F6). Set byte check #1 byte number (the **BC#1 byte number** parameter) to 1 – this tells the processor to check the first data byte in any note on message whose statusbyte matched our operation's message type (we chose 144). In this case, the 1st data byte of a note on message is the note number.
- set byte check #1's **check operand** parameter to the 'is in range' condition, the **low value** parameter to 0, and **high value** parameter to 59. This specifies that any note value below middle C will fit this condition.
- set byte check #2's **byte number** parameter to 2 (the note velocity data byte), the **check operand** parameter to 'is in range', the **low value** parameter to 1 and the **high value** parameter to 127. By doing this, we have provided the additional condition that only note on messages with a velocity higher than zero will be accepted for transformation (in this case, sent for output on midi channel 4).
- go to the transform parameters menu (F7), set the transform #1 byte number (the **X#1 byte number** parameter) to zero (the status byte), set the **transform operand** parameter to the 'set to' function, and type 147 into the **transform value** parameter. This first transform changes the statusbyte of all midi messages that matched our two byte checks to 147 – a note on message on channel 4.

Now we have to create a second operation to convert note on messages with a velocity of zero under middle C into actual note off messages, and transmit them to channel 4 to avoid stuck notes.

- Press [alt] – [c] to copy the current operation.
- go back to the operations list by pressing F8 twice. Move the cursor over the next empty operation slot. Press [alt] – [p] to paste the operation we just created and copied into this new slot. Select our new operation, enter the operation parameters menu, and change the **operation name** to 'note on v. zero'.
- go to the bytecheck parameters menu [F6], and change byte check #2's **byte number**, **check operand**, and **low value** and **high value** parameters to 2, 'is equal to', 0, and 0 respectively. Leave the first byte check as is (since we are still working with notes below middle C). The second byte check singles out all notes with a velocity of zero. Note that the high value parameter is unused when the check operand is in equality mode ('is equal to').
- go to the transform parameters menu [F7], and ensure that transform #1's **byte number**, **operand**, and **value** parameters are 0, 'set to', and 131 respectively. This transforms a note on velocity zero message from channel 1 into a note off message on channel 4.

Finally, we must implement an operation to send the appropriate incoming note off messages to channel 4 so as to avoid stuck notes that never receive a note off message.

- go back to the operations list by pressing F8 twice. Move the cursor over the next empty operation slot. Press [alt] – [p] to paste the note on operation still in memory. Select our new operation and enter the operation parameters menu. Change the **operation name** to 'note off ch. 1', the **message type** to 128 (note off channel 1), **# of byte checks** to 1 and **# of transforms** to 1. By setting the # of byte checks parameter to 1, we de-activate the second bytecheck which is not needed in this case. For clarity's sake, you might want to set the parameters for byte check #2 to zero.
- go to the transform parameters menu, and change transform #1's **byte number**, **operand**, and **value** to 0, 'set to', and 131 respectively. This transforms the note off message's statusbyte to a note off message on channel 4.

The midi processor is now programmed to perform a midi split on all incoming note data on midi channel 1. If it doesn't seem to be working, make sure the processor is turned on in the realtime settings menu (by pressing [alt-escape] followed by [F6], or by pressing [F8] to back up through the menu hierarchy until the root menu appears).

From this example, you can see how additional transformations on the notes below middle C could be performed. For instance, a transform could be set up to transpose those valid note messages down an octave (or any value), add an offset to their velocity to ensure a minimum volume, or change a keyboard's velocity response curve by using the multiplication or division operands on the velocity data

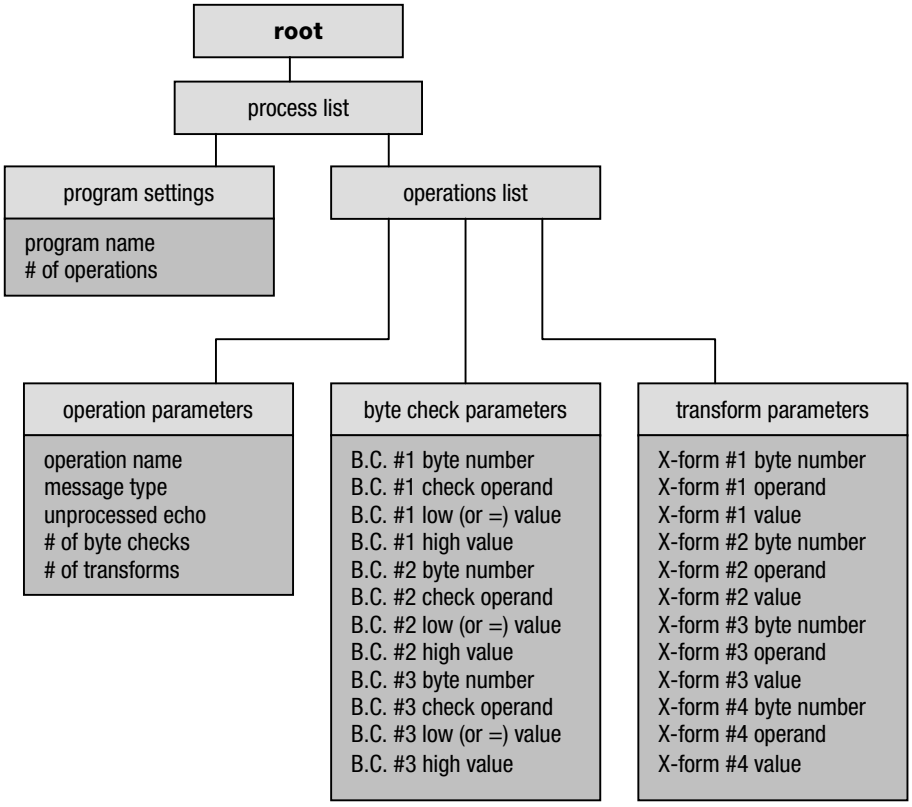
byte. Very complex midi stream manipulations can be achieved. Just remember that if you transform the note number or channel of note on messages, you must perform the same transformations to their respective note off messages - both actual note off messages and note on messages with a velocity of zero; otherwise, notes will get stuck on from never receiving a midi message telling them to turn off. Both the process list and operations list should be programmed sequentially from slot 1 to slot 12; in other words, it's not crucial but it's a good idea not to leave empty slots between active programs or operations, and to set the **number of operations** parameter (in the program settings menu) to the number of operations actually used in the respective program. This avoids unnecessary additional processing time and reduces the potential for mistakes when programming the midi processor.

The copy and paste feature was demonstrated in the midi split example above. Both operations as well as entire programs can be copied and pasted into other slots to speed up process list programming. From within either the process list or related program-specific menus, the selected program (including all its operations) can be copied by pressing [alt] – [c], and pasted by pressing [alt] – [p]. The same feature exists for copying individual operations; from either the operations list or from within any of the operations parameter pages, an operation can be copied using [alt] – [c] and pasted using [alt] – [p].

Note that incoming system real-time messages, such as clock, active sensing, system reset, etc. are not currently modifiable by the midi processor, even though their statusbytes can be selected. However, a non-system real-time message can be *converted into* a system real-time message by the midi input processor. Just make sure you use the 'xtend msg to' or 'crop msg after' operands to ensure the midi message length conforms to midi specifications – otherwise extraneous data bytes may be transmitted which could confuse a receiving midi device.

For additional information on the specifics of midi, as well as quick reference charts on the data formatting of various midi messages, refer to the midi appendix near the end of this manual.

The following is a structural map of the menu hierarchy for the midi input menu section (F2 – midi parameters), which may help familiarize the location of the various parameters.



program name
Edits the name for the selected processor program.

of operations
Selects the number of operations used in the program. All operations, byte checks, and transforms are performed sequentially starting with the first. This parameter allows you to disable the use of certain operations within the program; for instance if there are 6 operations and this parameter is set to 5, operation 6 will not be performed on the incoming midi data. This parameter should not be set to a value higher than the number of programmed operations, in order to both reduce processing time on slower computers and reduce the chance for possible user programming errors.

midi program operation parameters

operation name
Edits the name for the selected operation.

message type
Selects the statusbyte that identifies the messages to be processed by the current operation. Any midi message that begins with this statusbyte will pass through this operation (and be transformed if the additional 'bytechecks' (conditional verifications) are met). Possible values are between 128 and 255.

unprocessed echo
Enables or disables transmission of the original unprocessed version of the midi message (selected by the current operation and byte checks) to the midi output accompanied by the new processed message. This feature, as an example, would permit a layered effect if used in a midi split program such as described previously in the programming tutorial – simply enable unprocessed echo for all three of the programmed split operations, and the lower half of the keyboard will transmit on both channel one as well as four.

of byte checks
Defines the number of conditional verifications to perform on the incoming midi message. This parameter can be used to disable certain conditional verifications in the same way as the **# of operations** parameter in the midi program settings menu.

of transforms
Defines the number of transforms to perform on the incoming midi message. This parameter can be used to disable certain conditional verifications in the same way as the **# of operations** parameter in the midi program settings menu.

operation byte check parameters

Each operation has three independent byte checks (conditional verifiers), each of which can perform a comparison on any databyte of a midi message (even the same databyte as one of the other comparators). Because each operation has a dedicated byte check for the statusbyte (through the message type parameter in the operation parameters menu), these byte checks only reference the data bytes of a midi message. If more than one byte check is used, AND verification logic is applied. In other words, each byte check used must return a TRUE condition in order for the midi message in question to be processed by the operation. Each of the conditional verifiers have the following parameters:

byte check byte number (BC byte number)

Selects which databyte in the midi message to perform the conditional verification on. Possible values are between 1 and 99.

byte check operand

Selects what type of comparison to perform on the selected data byte. The possible selections are a range comparison, an equality comparison, and no comparison. The range comparison could be used to select note on messages of a certain range of the keyboard, or within a certain velocity range, for processing. The equality comparator could be used to select an individual note.

byte check low (or = to) value

Selects the low cutoff for the range comparator, or the value for the equality comparator. Possible values are between 0 and 127.

byte check high value

Selects the high cutoff for the range comparator; this parameter is unused for an equality comparison. Possible values are between 0 and 127.

operation transform parameters

Each operation has four independent transforms which are performed sequentially if the midi message meets the parameters of all of the active conditional verifiers (byte checks). The transforms can manipulate any byte of a midi message including the statusbyte. Each of the four transforms have the following parameters:

transform byte number (X byte number)

Selects which byte in the midi message to perform the transform on. Possible values are between 0 and 16, with 0 referencing the message's statusbyte and 1 –16 referencing data bytes.

transform operand

Selects what type of transform to perform on the selected byte. The possible selections are: **no xform**, **add**, **subtract**, **multiply**, **divide**, **set to**, **copy to**, **xtend msg to**, **crop msg after**, **do not transmit**, **invert**, **use value from,** and **function**. The **do not transmit** transform allows the filtration of certain midi messages in the stream (note that this operator does not affect an individual byte of a message, but the entire message itself – therefore the other parameters (transform value and transform byte number), as well as any other of the four transforms for the current operation, are unused when this operand is selected). The **use value from** operand allows the substitution of a value from one of the 26 sources in the modulation matrix for the selected byte (for example, the value from one of the LFO generators, a channel from the audio input, mouse input, or gameport input). In other words, if the **use value from** operand is used to determine the velocity value of note on messages on midi channel 1 via the joystick A X-axis input, by moving the joystick you will modulate or affect the velocity values of incoming midi messages as they are processed. Note that the modulation values taken from these external input sources are pre-arithmetic processor; in other words, even if the joystick axis has a particular arithmetic transform performed on it, the actual value used in the midi processor is the original (pre-arithmetically transformed) value.

The **function** operand is different from the above operands, in that it actually does not affect the midi data itself, but is sort of a branch function. Its use allows for a particular midi message to control an internal aspect of converter, for example starting and stopping converter's clock generator or permitting a synthesizer's damper pedal to act as a source for converter's tap tempo functionality. For certain functions, such as 'clock tempo +range', the midi message value pointed to by the **transform byte number** parameter above is used for the range portion of this function – therefore, make sure to set the **transform byte number** parameter for this particular operation to an appropriate databyte number for the particular midi message being used. Note that any preceding arithmetic operations which affect this value will affect the range sent to the desired function.

transform value

Selects the value for the transform. Refer to the table below for the relationships between the transform operand and transform value. Possible values for this parameter are between 0 and 255. Note that if the **transform byte number** parameter is set to zero (or "statusbyte") for the particular transform, values of 128 and above will be translated into text labels such as "note off [ch. 3]" instead of their numerical equivalent, assuming that the **translate statusbytes** option in the display settings menu has been enabled.

The chart on the following page outlines the details of each transform operand, and any eccentricities worth noting.

operand	explanation of operand and transform value
no xform	no action is performed on any of the data, so transform value is irrelevant.
add / subtract multiply / divide	the transform value is added to or subtracted from the selected midi message byte. the selected midi message byte is multiplied or divided by the transform value.
set to	the selected midi message byte is set to the transform value.
copy to	the selected midi message byte is copied to a new midi message byte number, specified by the transform value. If the message byte number to be copied to is greater than the number of bytes in the source midi message, the new midi message is automatically extended to include the new byte (therefore, there is no additional need to use the 'xtend msg to' operand below). A quirk to note is that any arithmetic performed on this byte should be performed before it is copied if the original message length is shorter than the new transformed midi message.
xtend msg to (extend message to)	a data byte (or more) is added to lengthen the number of bytes of a midi message being processed, where the transform value determines the midi message byte position of the generated byte, hence the length of the message. The generated byte value is set to 0, and any undefined bytes between the end of the unprocessed midi message and the generated byte position of the new (processed) message are automatically filled with a value of 0. The number of bytes that defines the message type is also increased, so further transforms can be performed on the new bytes. The **transform byte number** is unused directly; however the parameter must be set to a valid byte number for the unprocessed message – typically use statusbyte (0).
crop msg after	shortens the byte length of the entire processed midi message to (and including) the data byte number defined with the transform value. The **transform byte number** is unused; however the parameter must be set to a valid byte number for the unprocessed message as above.
do not xmit (do not transmit)	filters (does not pass to midi out) the entire midi message, processed or unprocessed – see the above description of the **transform operand** parameter for further explanation. This operand does not use a **transform value.** Set **transform byte number** to statusbyte.
invert	inverts databyte value; values of 0 become 127, 127 become 0, 30 become 97, etc.
use value from	uses a databyte value from one of the 26 sources in converter's modulation matrix, taken at the instant the incoming midi message is processed; the modulation sources are: • mouse x-axis • mouse y-axis • mouse button A • mouse button B • joystick A X-axis • joystick A Y-axis • joystick B X-axis • joystick B Y-axis • joystick A button 1 • joystick A button 2 • joystick B button 1 • joystick B button 2 • audio left low-pass channel • audio right low-pass channel • audio left main/band-pass channel • audio right main/band-pass channel • audio left high-pass channel • audio right high-pass channel • lfo – sine • lfo – triangle • lfo – sawtooth • lfo – square • lfo – sample & hold • lfo – 9 step • lfo – 5 step • lfo - asymmetrical

operand	explanation of operand and transform value
function	Performs an internal function within converter, for example, starting and/or stopping converter's midi clock generator, starting/retriggering one or all of the lfo generators, etc. Possible functions are: • **start/stop clock** – single source i/o for clock generator • **start clock** – starts clock without subsequent stop on next use • **stop clock** – opposite of above • **clk tempo +1** – clock tempo incremented by one • **clk tempo –1** – opposite of above • **clk tempo +range** – value taken from control source is used to sweep the clock generator's tempo upwards from the clock's base tempo • **clk tempo –range** – opposite of above • **tap tempo src** – source operates in the same manner as the insert key to define clock generator's tempo • **start all lfos** – starts the transmission of midi data (if programmed to do so) from all 8 lfo generators • **stop all lfos** – opposite of above • **i/o all lfos** – single source on/off for all 8 lfo generators • **start sine lfo** – starts the transmission of midi data (if programmed to do so) from the sine lfo generator • **stop sine lfo** – opposite of above • **i/o sine lfo** – single source on/off for the sine lfo generator • **start xxx lfo** etc…. – same as last three • **retrig all / xxx lfo** etc…. retriggers (starts at beginning of wave cycle) all lfos / a single lfo generator • **audio in on** – enables audio to midi conversion subsystem of converter. This includes all filter channels as well. • **audio in off** – opposite of above. • **audio in i/o** – single source on/off for the audio to midi conversion subsystem. • **L-LP thresh.** – gate threshold/trigger parameter for the left low pass audio channel. Each audio channel is represented in a similar way, such as R-HP meaning the right high pass channel, L-BP meaning the left main / band pass channel, etc. • **L-LP release** – the gate release parameter for the left low pass audio channel. • **L-LP dk time** – the gate decay time parameter for the left low pass audio channel. • **L-LP freq.** – the filter frequency parameter for the left low pass audio channel. • **L-LP Q** – the filter Q parameter for the left low pass audio channel. • **L-LP gain** – the filter's gain parameter for the left low pass audio channel.

gameport input

Gameport to midi conversion allows for a cheap and easy source of additional real-time controllers, and the simplicity of the interface makes it easy to design and build customized control surfaces (for instance, controllers for mixing or a machine control interface with data entry buttons or sliders, etc). The nature of the analog gameport is such that not just conventional controls such as knobs can be easily interfaced to it, but also such things as photoresistors (which could act as a light-oriented expression controller) and other variably-resistant devices. The standard analog PC gameport is designed to support two joysticks, both with 2 axis (channels which can be used for variable controllers) and two buttons (which can be used for on/off type controllers), for a total of 4 variable and 4 I/O controller devices. Sounds could be triggered from any of the gameport channels by programming them to transmit note-on messages with the conversion byte in the 2nd databyte of the message; according to the midi specification a note on message with a velocity of zero is the same as a note off message, therefore this application would work with devices which properly follow the midi specification.

For those individuals with electronics experience looking to build specialized interfaces for gameport to midi use, see the gameport interface appendix at the end of this manual. It is very elementary electronics, among the simplest type of electronics project to build, but please be careful – it is very possible to do real damage to the actual hardware components in your computer if you are not very careful, and we take absolutely no responsibility for your actions. Make sure to follow the pinout diagram / chart which applies to your particular 15-pin gameport connector – if you are using the connector on your soundcard, please pay attention to the specifics for that standard as opposed to the original IBM-standard pinout.

Please understand that the gameport axis channels were never designed to be used as input devices with accurate range resolution, and while every effort has been made to ensure converter provides the most useable and reliable gameport performance possible, it may be wise not to expect precise accuracy from the gameport axis inputs for critical applications.

Due to the (stupidly flawed) way in which the standard IBM-PC joystick interface was designed (back in the very early 1980's), there can be substantial differences in the circuitry between various interfaces. For this reason it is important to perform the calibration functions for each axis channel of your particular setup (joystick/controller and interface port). More information on this procedure is found in the 'software configuration page' in the earlier part of the manual, or on the 'Gameport Input Settings' page in this section. In addition, even though the gameport interface code in converter has been carefully designed to be the fastest and most efficient gameport input method possible (with an almost ridiculous amount of time and effort invested), incorporating effective dynamic acceleration and deceleration of port reading based on user activity and adaptive smoothing filters, the simple reality is that gameport input requires an obscenely generous amount of cpu time to acquire an accurate axis reading. Unlike conventional implementations of gameport input, converter does not disable interrupts during gameport input in order to guarantee reliable performance for other subsystems. Therefore, the time spent acquiring the axis timeout value from the gameport should never affect audio or midi input performance, as these take priority over gameport input. As a result, if an audio buffer size less than 72 bytes is used when converter is run in an audio input mode, gameport input is disabled until converter is quit and re-run with a larger audio buffer size (selectable in the file `hardware.cfg` – see the section detailing this file in the beginning of this manual). This is due to the fact that the smaller audio buffer sizes generate so many audio interrupts that converter is unable to complete the timing-based gameport input routines properly in order to acquire an accurate reading. Additionally, even if the audio buffer size is greater than the minimum required, midi (or a combination of midi and audio) interrupts may occur so frequently that converter will still be unable to provide accurate axis range results (buttons

will not be affected). This should only occur, however, under extreme circumstances. If it happens, either reduce the amount of midi input load, disable some of the audio filter channels, or forgo the use of gameport axis input – or use the gameport axis inputs as simple button (on / off) channels (using arithmetic operators to expand acquired axis values above 5 (as an example) to a number above 64).

The joystick buttons status is acquired in a much more elegant (simple) fashion, and do not create the same performance effects as reading the joystick axis. Therefore, all four joystick buttons can be used without incurring the computing load required by the axis by simply disabling all four gameport axis in the gameport input settings ([F4] – [F5]) and then enabling gameport input in the root real-time settings menu.

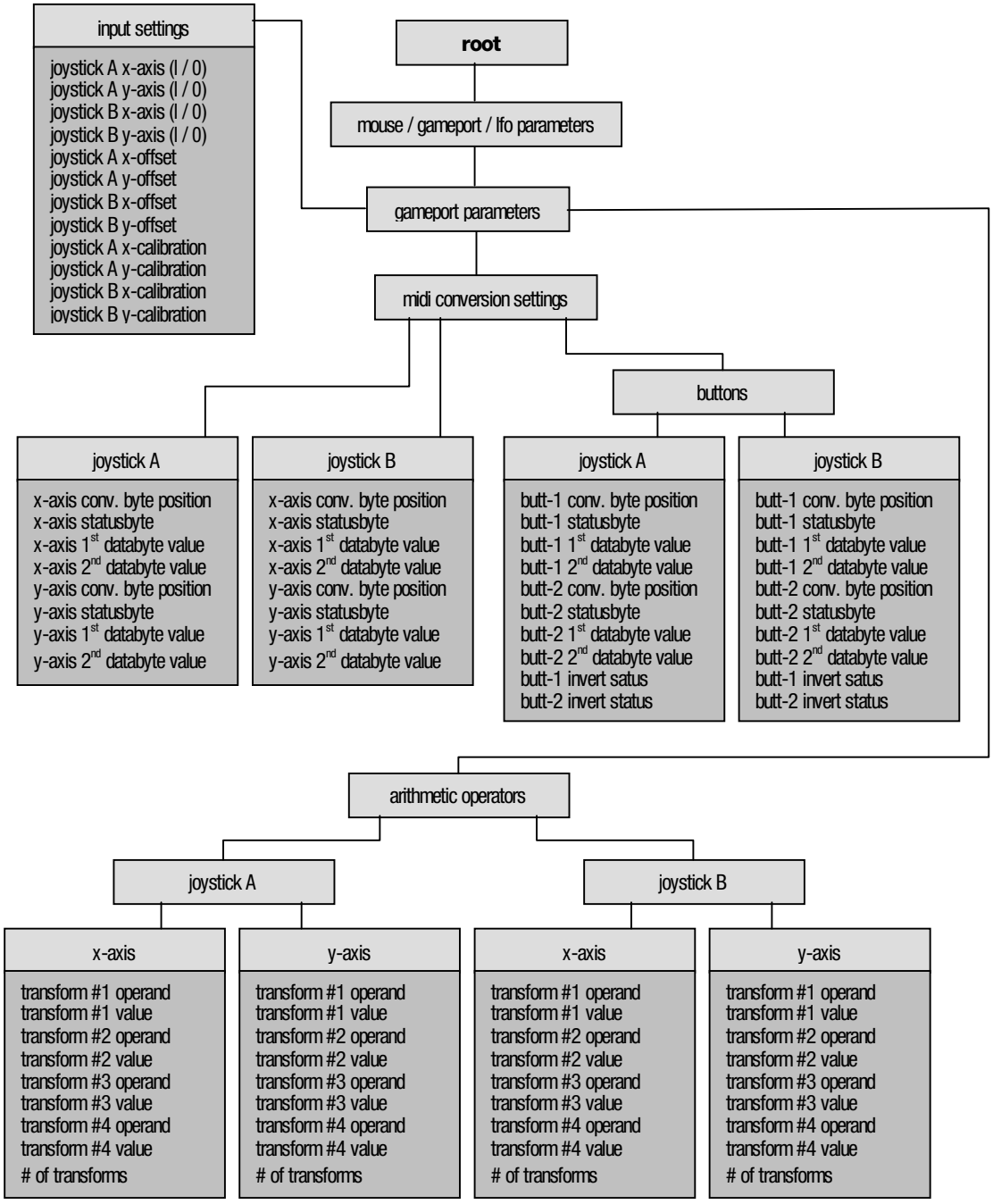
Programming gameport to midi conversion is simple – here's how to program joystick A to generate a stream of pitch bend and modulation wheel data on midi channel 1:

- Press [F4] for the game/mouse/lfo parameters menu, then [F5] for gameport settings, followed by [F5] again for gameport input settings. Check to make sure that the **joystick A x-axis** and **joystick A y-axis** parameters are set to "ON".
- Press [F6] for the gameport midi conversion settings menu, followed by [F5] for joystick A's settings page.
- For the **x-ax conv. byte pos.** parameter (x-axis conversion byte position) type 2 to place the converted byte value in the 2nd databyte position; for the **x-axis statusbyte** parameter type 224 (for pitch bend channel 1); followed by 0 for the **x-ax 1st databyte val** parameter (x-axis 1st databyte value) and perhaps for clarity's sake place a 0 in the **x-ax 2nd databyte val** parameter (which is going to be filled in by the byte taken from the x-axis channel of joystick A).
- Next, to program modulation wheel data, move the cursor to the **y-ax conv. byte pos** parameter and type 2, then type 176 into the **y-axis statusbyte** parameter to specifiy a controller message on channel 1, type 1 into the **y-ax 1st databyte val** parameter to specify modulation as the controller message type, and optionally place a 0 in the **y-ax 2nd databyte val** parameter.

And that's it – the gameport is programmed to transmit pitch bend and modulation data on midi channel 1. If it doesn't appear to be working, check the root realtime settings menu to ensure that the gameport input is activated. If it says "gameport disabled" you will have to exit converter and choose an audio input buffer size of 72 bytes or larger. Incidentally, due to the somewhat inaccurate nature of gameports, pitch control may not be the most musically appropriate use for them, even though smoothing algorhythms have been implemented to reduce errors – refer to the midi appendix for a list of possible controllers the gameports could be assigned to.

Note: Even though selectable as a message type, audio, gameport, and mouse input cannot be converted to system exclusive (sysex), mtc/smpte ¼ frame, or certain undefined system common/realtime message types.

The following is a structural map of the menu hierarchy for the gameport input menu section (F4 – gameport settings), which may help familiarize the location of the various parameters.



gameport input settings

joystick axis

Enables or disables input from the specific axis (X or Y) of the specific joystick (A or B). By individually enabling only the channels needed, and disabling unused channels, overall system speed increases.

joystick offset

Allows for an offset adjustment of the input value from the specified axis of the specific joystick. This value is added to the input value. Possible values are between –127 and 127 – typically this can be left at a setting of zero.

calibrate joy axis

As there is some electrical variance between different joystick interfaces, these functions permit machine-specific timing calibration for the joystick port. The acquired calibration settings are stored in converter's main system configuration file `c.cfg`, so that once properly calibrated, these functions will never have be performed again unless a different gameport device is used. Notice that the 'calibrate joy axis' selections are not parameters themselves; the calibration functions are accessed by pressing [enter] when one of the four channel's calibration routines is selected with the menu cursor.

By entering into one of the calibration functions, the normal operation of converter is temporarily halted (converter enters standby mode) until calibration is completed. There are three steps for calibrating each axis channel of the gameport: values must be acquired for both the minimum, center, and maximum positions for the axis channel. Detailed on-screen explanations will guide you through all three easy calibration steps.

Once the calibration steps are completed, test the range of your joystick or gameport device to see if the calibration settings are satisfactory. If not, simply repeat the calibration process for that axis or channel until the acquired calibration settings are as ideal as possible.

gameport axis midi conversion settings

axis conversion byte position

Selects which databyte of the chosen midi message the converted value is placed in. For example, to convert x-axis joystick input to control the pitch wheel, the generated message would consist of one statusbyte and 2 databytes as follows: the statusbyte (indicating a pitch wheel message on midi channel X), followed by the first databyte representing the LSB component of the pitch wheel message (unused in this case since we can only provide an 8-bit value, therefore should be set to a value of zero), followed by the second databyte representing the MSB value for pitch wheel. Hence you would type 2 or scroll the parameter until it says "databyte #2" (to indicate that the value taken from the gameport should be placed in the 2nd databyte of the midi message). Since the statusbyte is not considered a data byte, this parameter should be read in terms of databyte number, not byte number in the cumulative message packet (i.e. 2 instead of 3). Possible databyte position values are either 1 or 2; if set to zero (or "unused in msg"), the converted byte is not added to the midi message to be transmitted (in this way, a fixed midi message will be generated).

Note that not all midi messages use both, or any, data bytes; channel aftertouch, program change, and song select only use the first databyte (for aftertouch amount, program change number and song number respectively), while system realtime messages such as midi clock simply use the statusbyte. As converter automatically enforces these midi definitions, setting this byte position parameter to 2 when the message type only supports one data byte results in the converted value not being a part of the generated midi message.

axis statusbyte

Determines the midi message type (and channel) to use for the joystick conversion value. Possible values are between 128 and 255.

axis 1st databyte value

Sets a fixed value for the first databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 1st databyte position.

axis 2nd databyte value

Sets a fixed value for the second databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 2nd databyte position.

axis midi xmit

Determines whether the particular axis channel is used to generate midi data or not. This parameter is useful if using the gameport input as a controller for converter's own parameters (see section on control functions), and it is preferable that it not generate midi data.

gameport button midi conversion settings

button conversion byte position

Selects which databyte of the chosen midi message the converted value is placed in. For more information, refer to the midi conversion settings documentation for the joystick axis inputs.

button statusbyte

Determines the midi message type (and channel) to use for the joystick conversion value. Possible values are between 128 and 255.

button 1st databyte value

Sets a fixed value for the first databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 1st databyte position.

button 2nd databyte value

Sets a fixed value for the second databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 2nd databyte position.

button invert status

Enables or disables inversion of the button's input value. This allows the button to generate its direct on / off status as is, or to provide an off status when physically in the on position and an on status when physically in the off position.

button midi xmit

Determines whether the particular button is used to generate midi data or not. This parameter is useful if using the gameport input as a controller for converter's own parameters (see section on control functions), and it is preferable that it not generate midi data.

gameport arithmetic operators

Note that the arithmetic operations are performed sequentially in the order they appear in the transform list, from first to last – 'order of operations' is not followed. This allows for the input data to be processed in a more flexible manner.

transform operand (X-form)

Selects the type of data transform to be performed on the value converted from the respective joystick axis input. Possibilities are: no xform, add, subtract, multiply, divide, and invert.

transform value

Selects the numeric value for the transform. For instance, if the chosen transform is divide, and the transform value is 3, all converted values will be divided by three. Possible values are between 0 and 127 (protection is implemented against division by zero).

of transforms

Determines how many transforms are active, counting from the first transform. This makes it easy to bypass certain transforms and only activate the first 1, or 2, or 3 transforms. If set to zero, none of the transforms will be performed on the converted data.

joy-A x-axis function / joy-A y-axis function etc…
joy-A button 1 function / joy-A button 2 function etc….

Selects the type of function to be performed internally in converter when the associated button is pressed or movement is detected on one of the axis. These functions are performed separately from any midi conversion associated with the particular input source; for example, it is possible to have joystick A's button 1 both generate a midi start message as well as start converter's internal clock generator. For a complete list of the possible functions, please refer to the function listing on page 56 in the midi input processor section of this manual.

Certain functions make more sense if assigned to a binary (on/off) control source (such as one of the joystick buttons) rather than a range source (such as one of the joystick axis). If a function such as 'clock tempo range' is assigned to a joystick axis, that particular axis can be used to sweep converter's clock speed.

Note that the range value used for the functions is no longer taken post arithmetic processor as of version 1.4.

Note also that due to the relative inaccuracy of the gameport in general, certain functions may be better suited to either midi or mouse control.

mouse input

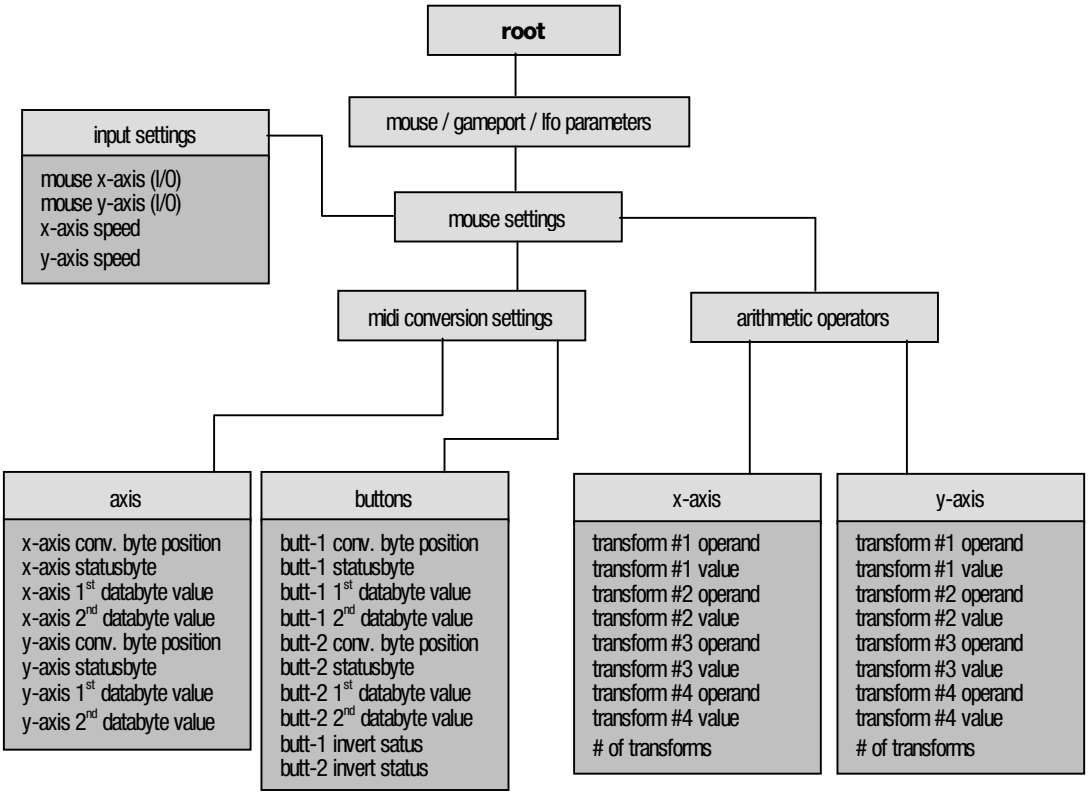
Mouse input conversion is programmed in the same way as the gameport input, so additional explanation would be redundant. Mouse to midi conversion was implemented to allow the use of certain special control surfaces, such as a touchpad, to be used as a controller source – in much the same way as a ribbon controller on some synthesizers. Since a touchpad device has 2 axis (X and Y), it can be more expressive than an ordinary one-dimensional ribbon controller, as two different controller types can be generated from the one pad.

If the mouse tracking speed is too slow or too fast, try adjusting the x-axis speed or y-axis speed parameters in the mouse input settings menu.

Note that any DOS-mouse-interface compatible device can be used, connected to either a serial or PS/2 mouse port (assuming the DOS mouse driver supports both the port and the device). This includes pretty much any pointing device that plugs into the serial or PS/2 mouse port. A serial-interface Cirque Glidepoint Trackpad was used for development. This device was detected and supported by the Logitech DOS mouse driver (version 7.2) on the development system with no problems.

Note: Even though selectable as a message type, audio, gameport, and mouse input cannot be converted to system exclusive (sysex), mtc/smpte ¼ frame, or certain undefined system common/realtime message types.

The following is a structural map of the menu hierarchy for the mouse input menu section (F4 – mouse settings), which may help familiarize the location of the various parameters.



mouse input settings

mouse x-axis
mouse y-axis

Enables or disables input from the specific axis (X or Y) of the mouse. This is useful if only one axis is needed for a particular application, and a second axis would interfere with the midi stream or generate unnecessary data.

x-axis speed
y-axis speed

Sets the tracking speed for the individual mouse input axis, allowing for small movements to cover a wide range of values, or large movements to cover a small range of values for finer control. Possible values are between 0 (slow tracking) and 127 (fast tracking).

mouse axis midi conversion settings

axis conversion byte position

Selects which databyte of the chosen midi message the converted value is placed in. For example, to convert x-axis mouse input to control the pitch wheel, the generated message would consist of one statusbyte and 2 databytes as follows: the statusbyte (indicating a pitch wheel message on midi channel X), followed by the first databyte representing the LSB component of the pitch wheel message (unused in this case since we can only provide an 8-bit value, therefore should be set to a value of zero), followed by the second databyte representing the MSB value for pitch wheel. Hence you would type 2 or scroll the parameter until it says "databyte #2" (to indicate that the value taken from the mouse should be placed in the 2nd databyte of the midi message). Since the statusbyte is not considered a data byte, this parameter should be read in terms of databyte number, not byte number in the cumulative message packet (i.e. 2 instead of 3). Possible databyte position values are either 1 or 2; if set to zero (or "unused in msg"), the converted byte is not added to the midi message to be transmitted (in this way, a fixed midi message will be generated).

Note that not all midi messages use both, or any, data bytes; channel aftertouch, program change, and song select only use the first databyte (for aftertouch amount, program change number and song number respectively), while system realtime messages such as midi clock simply use the statusbyte. As converter automatically enforces these midi definitions, setting this byte position parameter to 2 when the message type only supports one data byte results in the converted value not being a part of the generated midi message.

axis statusbyte

Determines the midi message type (and channel) to use for the mouse conversion value. Possible values are between 128 and 255. Note that certain midi message types are not allowed, simply because they are either undefined (such as 253 or 245), or are complex multi-byte messages such as 240 (system exclusive) or 241 (midi time code quarter frame).

axis 1st databyte value

Sets a fixed value for the first databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 1st databyte position.

axis 2nd databyte value

Sets a fixed value for the second databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 2nd databyte position.

axis midi xmit

Determines whether the particular mouse axis channel is used to generate midi data or not. This parameter is useful if using the mouse input as a controller for converter's own parameters (see section on control functions), and it is preferable that it not generate midi data.

mouse button midi conversion settings

button conversion byte position

Selects which databyte of the chosen midi message the converted value is placed in. For more information, refer to the midi conversion settings documentation for the mouse axis inputs.

button statusbyte

Determines the midi message type (and channel) to use for the mouse button conversion value. Possible values are between 128 and 255.

button 1st databyte value

Sets a fixed value for the first databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 1st databyte position.

button 2nd databyte value

Sets a fixed value for the second databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 2nd databyte position.

button invert status

Enables or disables inversion of the button's input value. This allows the button to generate its direct on / off status as is, or to provide an off status when physically in the on position and an on status when physically in the off position.

button midi xmit

Determines whether the particular axis channel is used to generate midi data or not. This parameter is useful if using the gameport input as a controller for converter's own parameters (see section on control functions), and it is preferable that it not generate midi data.

mouse arithmetic operators

Note that the arithmetic operations are performed sequentially in the order they appear in the transform list, from first to last – 'order of operations' is not followed. This allows for the input data to be processed in a more flexible manner.

transform operand (X-form)

Selects the type of data transform to be performed on the value converted from the respective mouse axis input. Possibilities are: no xform, add, subtract, multiply, divide, and invert.

transform value

Selects the numeric value for the transform. For instance, if the chosen transform is divide, and the transform value is 3, all converted values will be divided by three. Possible values are between 0 and 127 (protection is implemented against division by zero).

of transforms

Determines how many transforms are active, counting from the first transform. This makes it easy to bypass certain transforms and only activate the first 1, or 2, or 3 transforms. If set to zero, none of the transforms will be performed on the converted data.

mouse control functions

x-axis function / y-axis function / button 1 function / button 2 function

Selects the type of function to be performed internally in converter when the associated button is pressed or movement is detected on one of the axis. These functions are performed separately from any midi conversion associated with the particular input source; for example, it is possible to have the left mouse button both generate a midi start message as well as start converter's internal clock generator. For a complete list of the possible functions, please refer to the function listing on page 56 in the midi input processor section of this manual.

Certain functions make more sense if assigned to a binary (on/off) control source (such as one of the mouse buttons) rather than a range source (such as one of the mouse axis). If a function such as 'clock tempo range' is assigned to a mouse axis, that particular axis can be used to sweep converter's clock speed.

Note that the range value used for the functions is no longer taken post arithmetic processor as of version 1.4.

lfo generators

In addition to the previously described input sources, converter provides 8 lfo (low frequency oscillator) generators. Each lfo can be assigned to generate its own independent stream of midi continuous controller data, used as a modulation source for the midi input processor, or both. All 8 lfo generators are automatically synchronized to either incoming midi clock, converter's internal clock generator, or simply the tap-tempo time periods (if clock generation is undesired). Each lfo can be assigned its own individual note length and optional target midi message type to generate (as a continuous controller stream); additionally each lfo has its own arithmetic operator which can perform various operations to transform the output of the lfo. This all comes together to provide a very powerful and useful tool for using older hardware in a midi sequencing (or even live) environment.

Each lfo generator is hard-wired to a particular shape (sine wave, triangle, sawtooth, square, sample & hold, 9-step, 5-step, asymmetrical sine) but these shapes can easily be modified by using their arithmetic operators (for example, to turn the sine lfo into another square wave lfo, or the triangle lfo into another stepped lfo).

The transmission of midi message streams from the lfos are controlled individually from within the "transmit enable/disable" menu ([F4] – [F7] – [F5]). In this way, one or two lfo generators could be assigned to generate midi continuous controller message streams, while other lfos are only used as modulation sources within converter. The lfo generators as a whole source can be turned on or off within the root realtime menu ([Alt]-[E]), however under normal circumstances it is suggested they be left on as they do not require any significant cpu load.

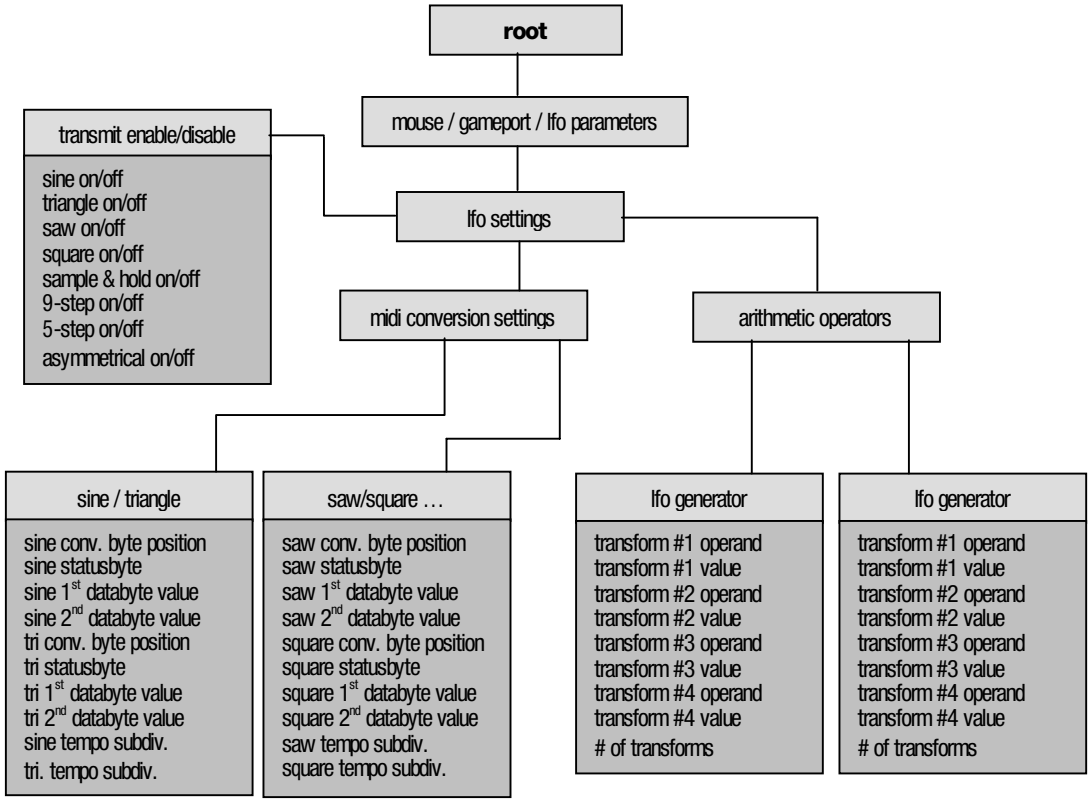
Programming the lfo generators is very simple, and is done in much the same way as the mouse and gameport input sources; hence all three sources are grouped into the same menu section ([F4]) .

Here's how to program the sinewave lfo to generate a stream of channel volume messages on midi channel 1:

- Press [F4] for the game/mouse/lfo parameters menu, then [F7] for lfo settings.
- Press [F6] for the lfo midi conversion settings menu, followed by [F5] for the sinewave/triangle lfo settings page.
- For the **sine LFO byte pos.** parameter type 2 to place the converted byte value from the lfo generator in the 2nd databyte position of the midi messages; for the **sine LFO statusbyte** parameter type 176 (for a controller message on channel 1, or scroll using the page up/page down keys until the parameter displays **cntrl [ch. 1]**); followed by 7 for the **sine 1st databyte val** parameter (indicating channel volume as the controller message type) and perhaps for clarity's sake place a 0 in the **sine 2nd databyte val** parameter (which is going to be filled in by the byte taken from the sine wave lfo generator).
- Next, move the cursor to the **sine tempo subdiv.** parameter and set the parameter to ¼ note.
- Finally, enable transmission of the lfo by pressing [F4], [F7], then [F5], and turn on the transmission of the sine LFO to begin sending a stream of midi messages.

And that's it – the sine lfo is generating (clock-sync-able) continuous controller messages on midi channel 1. If it doesn't appear to be working, check the root realtime settings menu to ensure that the lfo generators are activated (if you see the lfo lights pulsing, they are already enabled).

Here's a basic outline of the menu hierarchy for the lfo generators.



lfo transmit enable/disable

sine LFO / triangle LFO / saw LFO …

Enables or disables midi continuous controller message transmission from the particular lfo generator selected. By disabling transmission of one or all of the lfo generators, their usage as a modulation source within converter is not affected (unless the lfos are globally disabled from within the root realtime engine menu).

lfo midi conversion settings

lfo byte position

Selects which databyte of the chosen midi message the converted value is placed in. For example, to convert the value from the sine lfo generator to control the pitch wheel, the generated message would consist of one statusbyte and 2 databytes as follows: the statusbyte (indicating a pitch wheel message on midi channel X), followed by the first databyte representing the LSB component of the pitch wheel message (unused in this case since we can only provide an 8-bit value, therefore should be set to a value of zero), followed by the second databyte representing the MSB value for pitch wheel. Hence you would type 2 or scroll the parameter until it says "databyte #2" (to indicate that the value taken from the lfo generator should be placed in the 2nd databyte of the midi message). Since the statusbyte is not considered a data byte, this parameter should be read in terms of databyte number, not byte number in the cumulative message packet (i.e. 2 instead of 3). Possible databyte position values are either 1 or 2; if set to zero (or "unused in msg"), the converted byte is not added to the midi message to be transmitted (in this way, a fixed midi message will be generated every time the lfo generates a byte).

Note that not all midi messages use both, or any, data bytes; channel aftertouch, program change, and song select only use the first databyte (for aftertouch amount, program change number and song number respectively), while system realtime messages such as midi clock simply use the statusbyte. As converter automatically enforces these midi definitions, setting this byte position parameter to 2 when the message type only supports one data byte results in the converted value not being a part of the generated midi message.

lfo statusbyte

Determines the midi message type (and channel) to use for the lfo conversion value. Possible values are between 128 and 255. Note that certain midi message types are not allowed, simply because they are either undefined (such as 253 or 245), or are complex multi-byte messages such as 240 (system exclusive) or 241 (midi time code quarter frame).

lfo 1st databyte value

Sets a fixed value for the first databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 1st databyte position.

lfo 2nd databyte value

Sets a fixed value for the second databyte of a midi message. The **conversion byte position** parameter overrides this value if it is set to place the conversion result in the 2nd databyte position.

lfo tempo subdivision

Specifies the note length of a half-cycle of the lfo wave. In other words, if the square wave lfo is set to a ¼ note subdivision, the waveform if high (maximum amplitude) for 1 quarter note length, and then low (minimum amplitude) for the next quarter note, and so on. Triplet values are supported.

lfo key retrigger

Determines whether the lfo wave is returned to the beginning of its waveform cycle when a (post midi input processor, if enabled) note on message (on the same midi channel as the lfo is programmed to transmit on) is received via the midi input. This allows for quick resynchronization of the lfo cycle, and fun changes to the rhythmic placement of the lfo in relation to whatever song is being worked with. This does not affect the lfo if it is not transmitting its own midi message (is only used as a modulator).

lfo arithmetic operators

Note that the arithmetic operations are performed sequentially in the order they appear in the transform list, from first to last – 'order of operations' is not followed. This allows for the input data to be processed in a more flexible manner.

transform operand (X-form)

Selects the type of data transform to be performed on the value converted from the respective lfo generator input. Possibilities are: no xform, add, subtract, multiply, divide, and invert.

transform value

Selects the numeric value for the transform. For instance, if the chosen transform is divide, and the transform value is 3, all converted values will be divided by three. Possible values are between 0 and 127 (protection is implemented against division by zero).

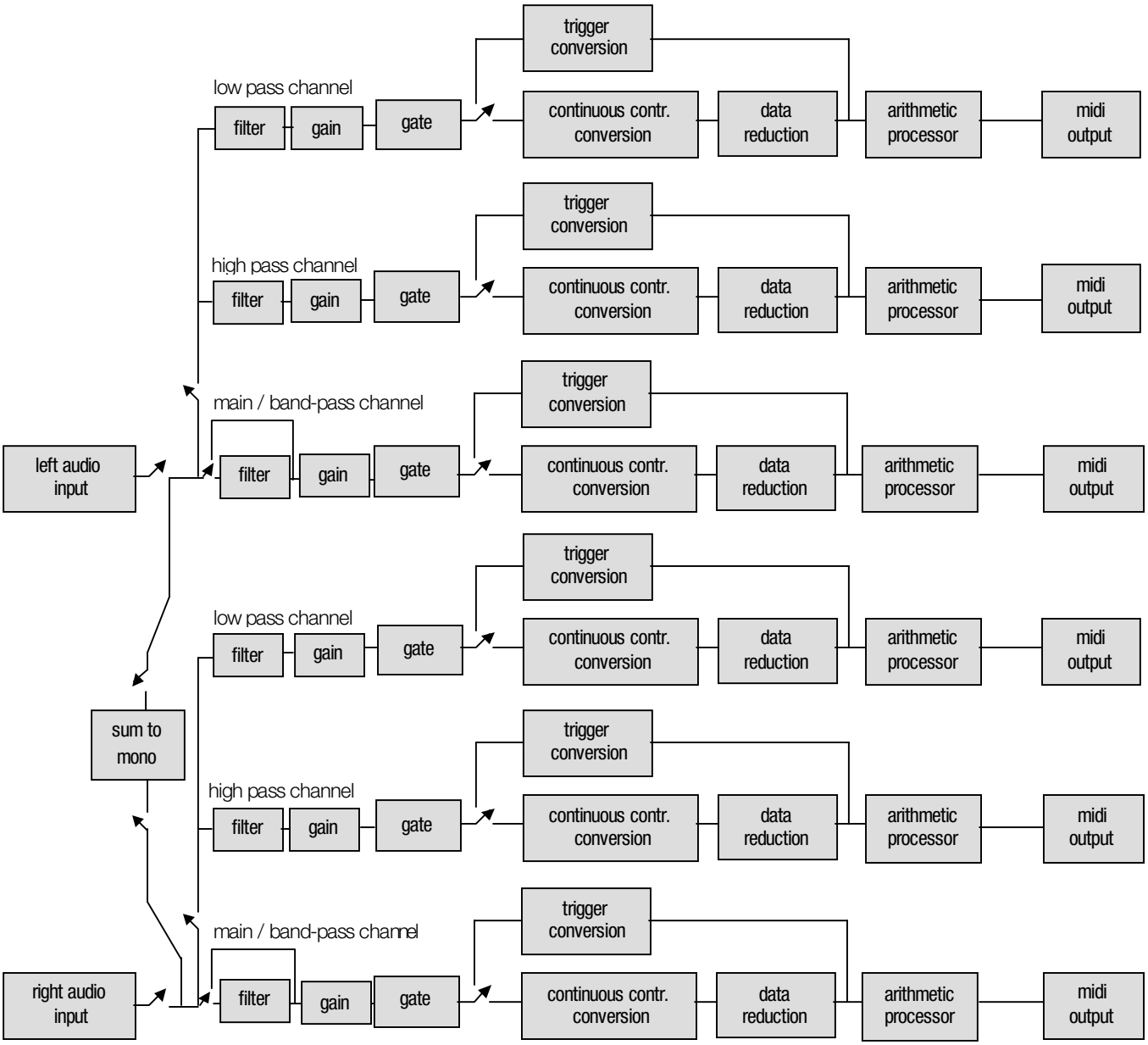
of transforms

Determines how many transforms are active, counting from the first transform. This makes it easy to bypass certain transforms and only activate the first 1, or 2, or 3 transforms. If set to zero, none of the transforms will be performed on the converted data.

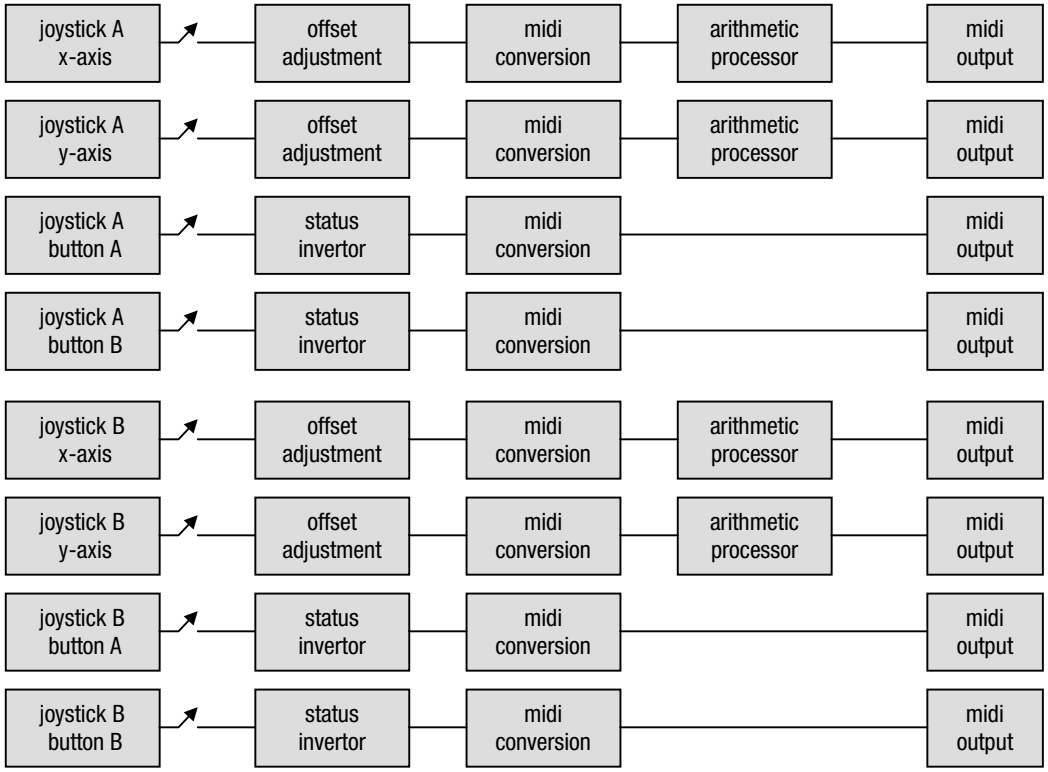
signal paths

The following diagrams show a simplified overall view of the flow of input sources to midi output within converter, and may help in understanding how certain settings affect other components within converter. Certain elements, such as the noise-shaping dc filters, gameport interpolation / smoothing filters, and other fixed / hardwired components in the signal flow have been omitted as their parameters are non-adjustable. If a more detailed explanation is required, drop us an email.

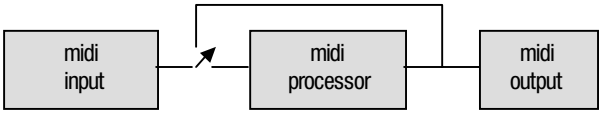
audio input signal paths



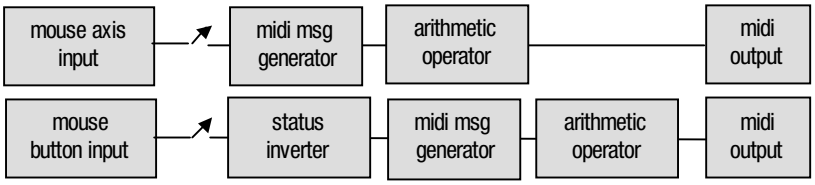
joystick input signal paths



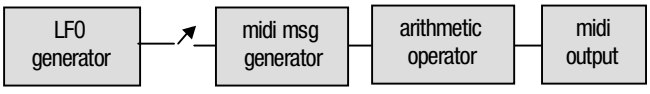
midi input signal path



mouse input signal path



lfo signal path



preset program files

Several 'preset' settings files (.set) accompany the converter distribution. These presets are pre-programmed for some of the typical applications of converter, and serve as templates which can be edited or tweaked to accommodate the specifics needed for a particular use.

Note that the file `default.set` is the work file converter uses to automatically 'remember' its settings between sessions. This means that converter can be quit, and then re-run, and all the settings will be restored from the last session – kind of like a hardware battery backup. If desired, converter can be reset to its startup settings by loading `default.set` at any time.

The file `empty.set` can be loaded to start or return converter to commonly used default settings with an empty midi input processor and all input sources turned off, and works somewhat like a 'new file' command.

The file `lastset.set` provides a safety backup of the program in use (edited or not) before a new program is loaded, helping to ensure any programming work is not accidentally lost.

midi input processor presets

The file `m-opslib.set` contains 132 individual operations, grouped into 11 different categories (or types) of midi input processing tasks. Each category of operations is stored in its own program slot in the midi input processor. This preset file for the midi input processor is not intended to be used directly 'as is', but instead to serve as a library of 'building blocks' which can be copied and pasted into your own programs.

All the operations are designed to process incoming midi messages on channel 1. Most of the operations can be used individually; the exceptions to this are the operations in the 'keybrd. split & xpose' (keyboard split and transpose) program slot – these operations are paired with partner 'note off' message operations, and must be used together in order to avoid stuck notes in all midi environments.

All of the operation slots in all the program slots are disabled, with the exception of the 1st slot titled "PASTE OPS IN HERE". This first slot can be used to build a custom program (group of operations) which can then be copied and pasted to a different '.set' program, or to simply try out the various preset operations held in the other program slots. Just copy an operation from one of the other groups and paste it in one of the operation slots in this first program.

audio input presets

There are 5 different audio to midi trigger program files, programmed for beat extraction from a variety of styles of music. These files should serve as useful templates which can be quickly tweaked or adjusted to work well with for a specific audio source. Some of the preset programs, such as `dltrig5.set` (programmed for a typical house track) illustrate the use of the audio to midi trigger decay time parameter to force certain note lengths (in this case, quarter note lengths for the bass drum etc). This technique can be useful for extracting rhythmic elements from a dense mix, where complex rhythm extraction is not as important as accurate tracking results.

midi data protocol specifications

Midi is a simple asynchronous serial interface able to transmit data at a rate of 31.25Kbps, which allows a maximum transfer rate of about 3000 bytes per second. Midi data is formatted into data 'packets' called messages. Each message begins with a status byte, and may be followed by any number of additional data bytes as determined by the message type. Some messages only consist of a statusbyte (notably the system realtime messages such as the midi timing clock), while others are followed by one, two, or more data bytes. For most applications using converter, only channel-oriented midi messages (called voice messages) will be used under normal circumstances (so don't worry about the variable multi-byte system exclusive messages, time code quarter frames, and the like unless necessary).

Midi data is always transmitted sequentially from statusbyte to last databyte. To reduce the transmission of redundant data, a type of run-length-encoding called running status exists, where messages of the same status byte are sent without re-sending the status byte. For instance, if three notes were pressed on a keyboard transmitting on midi channel one, there would be one status byte for a note on message on channel 1 (status byte 144), followed by the first note number and note velocity, then the second note number and velocity, and then the third note number and velocity. It is assumed that any data following that original status byte is the data bytes for an additional note on, and this condition remains until a new status byte is received which cancels this running status.

For a much more in-depth explanation of midi, do some research on the web, or visit:
`http://www.harmony-central.com/MIDI/Doc/doc.html`

voice messages (channel specific)

message type	status byte	data byte 1	data byte 2
note off	128 – 143	note number	note velocity
note on	144 – 159	note number	note velocity
polyphonic aftertouch	160 – 175	note number	aftertouch amount
controller / mode change*	176 – 191	controller number*	controller value*
program change	192 – 207	program number	not used
channel aftertouch	208 – 223	aftertouch amount	not used
pitch wheel control	224 – 239	pitch wheel LSB	pitch wheel MSB

* see the table of controller (or 'continuous controller') definitions on the next page.

system common and real-time messages (not channel-specific)

message type	status byte	data following statusbyte
system exclusive	240	machine – specific variable – length bytestream
midi time code ¼ frame	241	1 data byte, many bytes comb. for time position
song position pointer	242	byte #1: LSB byte #2: MSB
song select	243	1 data byte: song number
tune request	246	none
end of system exclusive (EOX)	247	none
timing clock	248	none
midi tick	249	none
start	250	none
continue	251	none
stop	252	none
active sensing	254	none
system reset	255	none

controllers / mode changes (MSB)

controller	controller number	data byte 2
bank select	0	value (MSB 0-127)
modulation wheel	1	value (MSB 0-127)
breath control	2	value (MSB 0-127)
foot controller	4	value (MSB 0-127)
portamento time	5	value (MSB 0-127)
data entry	6	value (MSB 0-127)
volume	7	value (MSB 0-127)
balance	8	value (MSB 0-127)
pan position	10	value (MSB 0-127)
expression	11	value (MSB 0-127)
effect control 1	12	value (MSB 0-127)
effect control 2	13	value (MSB 0-127)
general purpose controller 1	16	value (MSB 0-127)
general purpose controller 2	17	value (MSB 0-127)
general purpose controller 3	18	value (MSB 0-127)
general purpose controller 4	19	value (MSB 0-127)
damper pedal (on/off)	64	0 – 63 = off, 64-127 = on
portamento (on/off)	65	0 – 63 = off, 64-127 = on
sustenuto (on/off)	66	0 – 63 = off, 64-127 = on
soft pedal (on/off)	67	0 – 63 = off, 64-127 = on
legato footswitch (on/off)	68	0 – 63 = off, 64-127 = on
hold 2 (on/off)	69	0 – 63 = off, 64-127 = on
sound controller 1 (sound variation)	70	value (0-127)
sound controller 2 (timbre)	71	value (0-127)
sound controller 3 (release time)	72	value (0-127)
sound controller 4 (attack time)	73	value (0-127)
sound controller 5 (brightness)	74	value (0-127)
sound controller 6	75	value (0-127)
sound controller 7	76	value (0-127)
sound controller 8	77	value (0-127)
sound controller 9	78	value (0-127)
sound controller 10	79	value (0-127)
general purpose controller 5	80	value (0-127)
general purpose controller 6	81	value (0-127)
general purpose controller 7	82	value (0-127)
general purpose controller 8	83	value (0-127)
portamento control	84	source note
effects 1 depth	91	value (0-127)
effects 2 depth (tremulo)	92	value (0-127)
effects 3 depth (chorus)	93	value (0-127)
effects 4 depth (celeste/detune)	94	value (0-127)
effects 5 depth (phaser)	95	value (0-127)
data entry +1	96	none
data entry –1	97	none
non-registered parameter number LSB	98	value (LSB 0-127)
non-registered parameter number MSB	99	value (MSB0-127)
registered parameter number LSB	100	value (LSB 0-127)
registered parameter number MSB	101	value (MSB 0-127)

Note that the above list represents **defined controllers** according to the Midi Specification. Some of the gaps in the numbering represent numbers for which controllers have not yet been defined (hence in converter these values are referred to as **undefined controllers**) – they may be used, however most (or possibly all) midi gear will not respond to undefined midi controller messages.

high-resolution component (LSB) for controllers (unused by converter)

controller	controller number	data byte 2
bank select	32	fine value (LSB 0-127)
modulation wheel	33	fine value (LSB 0-127)
breath control	34	fine value (LSB 0-127)
foot controller	36	fine value (LSB 0-127)
portamento time	37	fine value (LSB 0-127)
data entry	38	fine value (LSB 0-127)
channel volume	39	fine value (LSB 0-127)
balance	40	fine value (LSB 0-127)
pan	42	fine value (LSB 0-127)
expression controller	43	fine value (LSB 0-127)
effect control 1	44	fine value (LSB 0-127)
effect control 2	45	fine value (LSB 0-127)
general purpose controller #1	48	fine value (LSB 0-127)
general purpose controller #2	49	fine value (LSB 0-127)
general purpose controller #3	50	fine value (LSB 0-127)
general purpose controller #4	51	fine value (LSB 0-127)

special controllers

controller	controller number	data byte 2
all sound off	120	0
reset all controllers	121	0
local control (on/off)	122	0 = off, 127 = on
all notes off	123	0
omni mode off & all notes off	124	0
omni mode on & all notes off	125	0
poly mode (on/off) all notes off	126	number of channels
poly mode on & all notes off	127	0

midi note numbers

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

As a general troubleshooting tip, if converter is having trouble functioning properly, or it is difficult to isolate what is interfering with its successful operation, try making a bootable floppy disk with converter installed on it – this may help determine if the problem is a memory, hardware or software issue.

loading converter

My soundcard isn't found by converter! or **converter crashes on loading!**

Make sure that the appropriate DOS environment variable for your soundcard is included in your `autoexec.bat` file – converter references the hardware information included in this string of text in order to locate your soundcard. See the appendix on "soundcard configuration" in this appendix for soundcard-specific information. The best approach is to use the installation software that came with your soundcard (or that is available on your soundcard manufacturer's website) to properly install and configure your soundcard on your PC.

If the environment variable is in your `autoexec.bat` file, it appears to be correct, but you are using a soundcard that implements emulation of a SoundBlaster 16 or SoundBlaster Pro, it may be that your soundcard isn't 100% compatible with the SoundBlaster 16 or Pro standard; or it may be a soundcard which doesn't implement audio or midi input under DOS, just audio and midi output (for use in games only, such as the PCI SoundBlasters from Creative Labs). The only solution in this situation is to use a different soundcard.

Another possibility is that there is not enough free memory available under DOS for converter – converter requires about 560k of free base memory under DOS. If there are a lot of memory-resident programs loaded at bootup (ie. from the `autoexec.bat` file), these may be taking up memory converter requires to function properly. If this is the case, try making a bootable floppy with converter and its own autoexec.bat file – see the section in this manual which describes this procedure for further information.

Check to make sure that the settings in the `hardware.cfg` file in converter's folder are correct and valid for your particular sound card. Also, make sure all the required files are in the same folder as the converter application (c.exe). If necessary, verify the files in the directory with the file listing earlier in this manual. If a file is missing, it is necessary to reinstall from the converter source archive, or re-download the converter zip archive from our website (www.urr.ca).

Finally, it may also be possible that the video card in your machine is not compatible with converter. To verify this, set the soundcard parameter in the file `hardware.cfg` to '10', to select 'no soundcard'. When run, converter will simply load into memory without initializing any sound interface. If converter still crashes, it is possible that your video card is too new (such as certain video cards manufactured after early 2002) or is just unsupported (rare, but possible).

My Roland MPU-401 (or compatible) isn't found by converter; instead the computer hangs after the software says "Initializing MPU-401…"!

This means you have specified the incorrect base address for the card in the `hardware.cfg` file. Check your card's jumper settings and/or try another address. If the base address setting in `hardware.cfg` is indeed correct, you may have another device installed in the computer which is using the same address as your MPU-401 – try to identify what that piece of hardware would be, or change the base address of your MPU-401 card to a different setting such as 300.

When I try to run converter, it crashes, and I know my soundcard isn't the problem!

Make sure you have enough free memory available in the base 640k that DOS uses – you can check by typing 'mem' at the DOS prompt. Memory above the base 640k is not directly available to converter, so even if a computer has been expanded to 128MB of RAM, converter doesn't make use of any of it (ordinarily it hasn't any need to). Since converter (and many other DOS-based applications) requires at least 560k of free ram , it's a good idea not to have a huge number of memory-resident software running at the same time if they are taking up a significant amount of base memory, such as DOS cd-rom drivers, EMM386, etc. A good workaround is to make a bootable floppy disk from which to run converter.

When I try to run converter, I get a message saying "Program too big to fit in memory"

Make sure you have enough free conventional memory available to converter – at least 560k of the base 640k of ram. It is best to not have EMM386 or some other protected mode driver loaded. Also, for the mouse driver, try using the following command in the `autoexec.bat` file to load the mouse driver into high memory:

```
loadhigh=mouse
```

When I try to run converter, it exits before anything happens, with the message "abnormal program termination". What gives?

This error is most likely caused by too little 'base' memory available for converter – for solutions, see the answer to the previous question.

Why won't converter run in a DOS shell under Windows?

During testing, it was found that converter didn't function predictably when run in a DOS shell (DOS prompt) under Windows9x. This is likely due to the fact that many of the computer's built in devices are controlled directly from converter itself, which may be interfering with Window's controlling nature in regards to hardware. Solution? Boot your machine into 'pure' DOS mode in one of several ways:

- Re-boot your computer, press F8 as Windows9x begins to boot to bring up a 'boot menu', and select 'command prompt only'
- create a bootable floppy and run converter from it (easy approach)
- reboot your machine into pure DOS mode by clicking on the start menu and selecting shutdown, and 'Reboot into MS-DOS'. Note that this approach may not automatically run your `autoexec.bat` file, which may be needed if you are using a SoundBlaster or Gravis Ultrasound card. If so, you can run the file yourself from the dos prompt by simply moving to your hard disk's root directory (ie. `c:\`) and typing `autoexec` followed by return.

Running converter in a DOS shell from within Windows9x would in effect reduce the benefits of it being designed for DOS in the first place, as the additional weight of Windows running in the background

would significantly reduce converter's realtime performance. For this reason, making it function properly from within Windows is not a priority.

When I try to run converter under Windows Millenium Edition, WindowsXP, or various flavours of Windows NT (including Windows2000), it doesn't work.

Although converter hasn't been tested under these platforms, it is automatically assumed that it will not function at all, due to the fact that these versions of Windows use a DOS emulator which does not permit any program to communicate directly with the computer's hardware. Since converter is almost entirely based on direct hardware communication specifically for performance benefits, it obviously will not get along with any of these Windows platforms. However, all is not lost – simply create a bootable floppy disk on a computer with Windows98 or earlier, copy the converter files onto the disk, and configure it for your computer's soundcard. By using a bootable floppy disk you don't need to worry about incompatibilities with your version of Windows since DOS is loaded right from the floppy. For more information see the appendix on creating a bootable DOS floppy disk.

hardware / interface issues

My Roland MPU-401 (or compatible) is detected and initialized by converter, however no midi data is received by the software!

This means you have specified the incorrect interrupt (IRQ) for the card in the `hardware.cfg` file. If the IRQ setting in `hardware.cfg` is known to be correct, another hardware device installed in your computer may be either using the same IRQ number. Try configuring your MPU-401 to use a different IRQ number, or locate the conflicting piece of hardware and re-configure it.

My SoundBlaster card is found and initialized, but audio input stops a few buffer frames after a signal is fed into the card!

This was originally an issue on the development machine, where the AWE64 was configured to use DMA channel 0 as its 8-bit DMA channel. Channel 0 is not typically recommended for use by peripherals, as historically this channel has been used for motherboard memory refresh cycles, and while that may no longer be the case on certain motherboards, and the SoundBlaster configuration utility may claim that the channel is free to be used, it may in fact be used by another device on the motherboard. In short, the DMA controller is re-programmed by that other device, taking control from converter and the SoundBlaster card, and stopping audio data input. During development, it was possible to 'force' this situation to work by re-programming the DMA controller chip each time an interrupt occurred from a complete buffer transfer, but this created significantly decreased system performance and also resulted in lost samples. The solution is to configure the card to use another DMA channel that is actually free – try channel 1 or 3.

using converter

I get the message "warning – CPU LOAD HIGH" – what should I do?

This message means that the computer has a very high task load and while no errors should be caused as a result, response times may be affected. Often when this error is displayed, converter's midi output data rate is very close to or exceeds the bandwidth of the midi interface specification, which will affect

the responsiveness of timing and such things. Either use the audio midi data reduction settings to reduce the amount of audio to midi conversion data (if using audio in continuous controller mode), turn off the gameport input, or otherwise reduce the amount of midi data and/or processing being performed by converter.

This situation should normally only occur under aggressive processing loads, fully utilizing most or all of converter's numerous conversion sources on slower computers.

I just booted up converter, and when I transmit midi messages from the external midi controller, converter appears to be receiving a corrupted midi stream (and/or the undefined data and data code display in the system status wedge flashes and shows data codes which are 'invalid')

This means that converter hasn't received a statusbyte yet from the midi input port, because the device sending the midi stream to converter is in running status mode and is assuming converter caught the initial statusbyte for the midi stream, which it hasn't (because it wasn't running or connected at that point). Without an initial statusbyte to tell converter what the data stream represents, converter will be unable to interpret and process the midi data. For example, if a midi controller was on and had been used (keys pressed or modulation wheel moved) before converter was loaded or connected to via the midi ports, converter missed the initial statusbyte that the controller had already transmitted to define its midi stream. This situation will happen more often with consumer midi controllers which only transmit a running status of note on messages without using the note off midi message to turn off voices (instead, note on velocity 0). To resend a statusbyte from the midi controller, either move the modulation or pitchbend wheel, press a patch select button, or (with more limited midi devices) turn off / turn on your midi controller.

When I program the midi processor to convert or transform a midi message, nothing happens!

Make sure that:

- converter is operating in a mode which enables midi input (see 'software configuration' for more information)
- the midi processor is enabled (in the realtime engine settings menu – [Alt]-[E])
- the # of operations parameter in the program setting menu for your midi processor program is set to the number of operations you've programmed, and not zero (which would mean none of the operations are performed)
- the # of byte checks and/or # of transforms parameter in the operation settings menu for the operation(s) you have programmed is set to the number of byte checks / transforms you are using for the operation, and not zero (which would mean none of the byte checks / transforms are performed)

When I use the 'do not xmit' operand in the midi processor, the message still gets transmitted!

See the last two points from the above question, concerning the # of operations in the program settings menu etc.

When I use unprocessed echo option in operation settings, I get stuck notes!

With note on and note off messages, it is important to have the proper number of note off messages to accompany the number of note on messages generated by converter; it is also important that the note off messages match the same note # and octave as the note on messages. The midi message texbox is a useful tool to see what midi messages are actually being transmitted, and therefore catch any programming errors. When unprocessed echo is used for an operation processing a note on message, the resulting output stream will contain two note on messages (even if they are for the same note # and octave). It is important, therefore, that two accompanying note off messages are transmitted to ensure

that both notes are actually turned off. This can be done by using another operation to process note off messages with the unprocessed echo option turned on as well (thereby creating the second note off message).

proper soundcard configuration under DOS

This section is provided for users configuring an older soundcard within DOS, for which there may be no accompanying documentation or installation software. **If you are experienced with DOS and setting environment variable hardware settings, or you have all the necessary software and documentation to properly configure the card under DOS, or your soundcard already works under DOS with other software (and especially if it works fine with converter), you will likely have no need to read this section.** If you have difficulties getting converter to locate your soundcard, this reference might be able to give you some pointers in the right direction. Note that the details of complete hardware installation for soundcards is far beyond the scope of what can be covered here; If there is a need for supportive files or documentation listed here which is no longer available on the web, drop an email to us and we will set up a support page on our website with the files for download.

In order for converter to make use of your soundcard, it must be able to find out its particular hardware settings which tend to differ from computer to computer based on the other peripherals one has installed in the machine. Both the Gravis Ultrasound and SoundBlaster series make use of what's called a 'DOS environment variable', or in other words a line of text placed in the **autoexec.bat** file (editable using Windows' notepad or DOS edit) which lists the specific information a software application needs in order to 'connect' with the interface. Both the Gravis Ultrasound and SoundBlaster cards were packaged with DOS-based configuration or installation utilities which allowed the setting (and testing) of correct hardware DMA, IRQ, and port address settings for the card. However, due to the fact that the soundcards have been discontinued, and the DOS-specific software for these 'obsolete' soundcards may not remain available on the manufacturer's website indefinitely, the following information is provided to assist in the successful configuration of these cards.

Note that this guide does not necessarily replace the need for the appropriate installation software and documentation for your particular card.

Note for users running with Windows9x: Even if you have the appropriate drivers installed for the card under Windows, you might want to verify that an environment string has been placed in the autoexec.bat file for DOS usage. Often it seems that the Windows-based installation software for soundcards doesn't automatically perform this task.

Some utilities to configure Plug & Play cards under DOS may require the "P&P OS Installed" parameter in the computer's BIOS to be set to NO or OFF.

DOS driver support page at our website - http://www.urr.ca/dosdrivers/drivers.htm

Gravis Ultrasound

The Gravis Ultrasound uses a jumper on the card itself to select the base address for the card (either 210, 220, 230, 240 etc), while the rest of the parameters (IRQ, DMA) are configured by software.

Environment variable: `SET ULTRASND=240,1,5,7,5`
The numbers after the = sign in the line represent:

```
base address, 8-bit DMA, 16-bit DMA, Audio (GF1) IRQ, Midi port IRQ.
```

Typically, the DOS software installation for this soundcard will automatically update the autoexec.bat file with the correct parameters. However, a caveat:

- if a Windows9x (specific) driver was installed as well, the ULTRASND string might instead look like this: `SET ULTRASND=XXX,X,X,X,X` (literally). To function under DOS, this line should be 'disabled' by typing the letters 'rem' at the beginning of the line (before the word 'SET'), and a new line with actual numerical information should be typed in to look something like this:

```
rem SET ULTRASND=XXX,X,X,X,X
SET ULTRASND=240,1,5,7,5
```

Note that the numbers above are for a specific configuration – they aren't 'one size fits all'. You must use the numbers specific to your system's configuration. **A bootable floppy could be created to run converter with its own autoexec.bat, thus avoiding this issue of modifying the autoexec.bat on your hard drive (see the next appendix).**

Assuming you have installed a Windows driver for the card (since you are reading this paragraph), you can find the specific numbers from the Windows driver by going to your control panel, selecting system, choosing the 'device manager' tab, expanding the 'sound, video and game controllers' droplist, and double-clicking the Gravis Ultrasound item. This should give you a window with several tabbed entries – click on 'resources'. Under the 'resource settings' listbox, read the following numbers and write them down on paper to be used for your environment string:

- the first number of the first entry (input/output range – this is your card's base address)
- the fourth and fifth entries (interrupt requests - these are the cards audio and midi IRQs)
- the fifth and sixth entries (direct memory access – these are the card's 8 and 16-bit DMA channels)

Place these numbers in their respective places on the DOS environment string (line of text) in your autoexec.bat file. The IRQ numbers can be interchanged, as well as the numbers for the dma channels – if they don't work for some reason, try switching them around.

SoundBlaster Pro, SoundBlaster 16, SoundBlaster AWE32, AWE64, & 100% compatible

The SoundBlaster series of soundcards have a variety of installation requirements – some are 'plug and play', others are not. The specifics of each soundcard's hardware configuration is beyond the scope of what can be covered here; however, often the card will be configured appropriately under Windows (if used) and the numbers for the environment variable can be sourced from it.

Environment variable: `SET BLASTER=A220 I10 D0 H7 P300 E620 T6`
The important numbers after the = sign in the line represent:

```
base address, IRQ, 8-bit DMA, 16-bit DMA, Midi port address
```

The E and T parameters are not important in this case, and can be omitted.

Typically, the DOS software installation for this soundcard (either ctcm.exe or diagnose.exe or the like) will automatically update the autoexec.bat file with the correct parameters. However, if this software isn't available, and the card has been configured successfully in Windows9x, the relevant information can be retrieved from the Windows driver.

Open the Windows9x control panel, double-click on 'system', choose the 'device manager' tab, expand the 'sound, video and game controllers' droplist, and double-click the appropriate Creative driver for *audio* (in our case, 'Creative AWE64 16-bit Audio (SB16 compatible)'). This should give you a window with several tabbed entries – click on 'resources'. Under the 'resource settings' listbox, write the following numbers down on paper to be used for your environment string:

- the first entry (interrupt request – this is your card's IRQ)
- the second and third entries (direct memory access – the one with a value of 3 or below is the SoundBlaster's 8-bit DMA channel, and the one with a value above 4 is the SoundBlaster's 16-bit DMA channel)
- the first number of the fourth entry (input/output rage – this should be the SoundBlaster's base address and have a value of something like 220, 240, 260, etc)
- the first number of the fifth entry (input/output range – this should be the SoundBlaster's MPU-401 midi port base address and should have a value of 300, 330, etc).

Place these numbers in their respective places on the DOS environment string (line of text) in your autoexec.bat file.

Roland UART-mode MPU-401 and compatible

These interface cards (generally) do not use a DOS environment variable to specify their base address and IRQ; therefore, the card's base address and IRQ are typed into the file `hardware.cfg` that converter uses for initialization (see the discussion on configuring this file earlier in this manual). Many of these interface cards (specifically the Roland cards, and many compatibles such as ones from Music Quest) used jumpers on the card to specify the base address and IRQ to be used. To determine the settings for your particular interface, either look at the interface card you have installed (and/or refer to the manual or documentation that came with the card), or if the card has been successfully installed under Windows9x refer to the card's settings under Windows as follows:

Open the Windows9x control panel, double-click on 'system', choose the 'device manager' tab, expand the 'sound, video and game controllers' droplist, and double-click the appropriate driver for your MPU-401 (in our case, 'MPU-401 Compatible' for our Roland MPU-IPC-T). This should give you a window with several tabbed entries – click on 'resources'. Under the 'resource settings' listbox, write the following numbers down on paper to be used in converter's `hardware.cfg` file:

- the first entry (input/output range – this is the base port address of the MPU card)
- the second entry (Interrupt Request – this is the IRQ your card uses)

Open Notepad (under Windows, or Edit under DOS) and place these numbers in their respective locations in the `hardware.cfg` file.

Soundcard manufacturer websites/webpages for manuals and drivers

Roland MPU-401, MPU-IPC-T, SCC-1, LAPC-1, etc.

Documentation available as HTML or PDF – scroll down until you find the section for MPU cards.

`http://www.rolandus.com/SUPPORT/DOCS/SUPNOTES.HTM`

Gravis Ultrasound Classic and Plug & Play series

Documentation, software installation, and utilities are available on this page. The soundcard sections are located towards the bottom of the list – try the following files:

- GUS411.ZIP (for Ultrasound Classic/Max series)
- P20DISK2.ZIP (for Ultrasound Plug&Play – may also require P20DISK1.ZIP but possibly not)

`http://www.gravis.com/support/sup_1059.html`

Creative Labs SoundBlaster Series – DOS/Legacy drivers

This particular Creative Labs website has various DOS-oriented utilities and installation disks available for download. Note that these downloads are not designed for non-Creative Labs hardware (ie. clones). Specific downloads to try are:

- "Basic DOS-level utilities for use in Windows95 MS-DOS mode or a Windows 95 Command Prompt only boot" (towards the top of the list – DOS software such as Diagnose.exe which can be used to configure your soundcard)
- "Sound Blaster 16/SB32/AWE32 Basic Disk for DOS/Windows 3.1 Installation"
- "Plug and Play Configuration Manager" (2 disks, towards the bottom of the list, for the Plug and Play versions of the SoundBlaster cards such as AWE-64, etc)

creating a bootable DOS floppy to auto-run converter

Since converter and its associated files take up relatively little disk space, the application can be run from a single 1.44MB 3.5" floppy disk. This fact enables converter to become a 'portable' application that can be run 'out of the box' so to speak by creating a bootable DOS disk that boots your computer into 'true' DOS mode and auto-loads converter without intervention. Using converter becomes as easy as placing the floppy in your disk drive and rebooting – very useful for Windows9x users who do not want to modify the autoexec.bat file on their computer's hard drive.

Here's the steps:

- Find a blank 1.44MB 3.5" floppy disk, and format it as a bootable disk in one of the following ways:
    - **Under Windows9x Explorer shell:** Insert the blank disk in the disk drive, double-click "my computer", select your floppy drive (typically A:), click on the File menu and select format. In the dialog box that appears, make sure "1.44Mb (3.5")" is selected under the "Capacity" heading, then select "Full" under "Format type", and select "Copy system files" under "Other options". If desired, a label for the disk can be entered in the "Label" textbox (completely optional). Click on "Start" to begin the process, which will take a minute or so.
    - **Under DOS Command Prompt / 'True' DOS mode:** Insert the blank disk in the disk drive, type `format a: /s` (3.5" floppy drives are typically a: under DOS) and press enter, and then enter again at the "..press enter when ready.." prompt that appears. The disk will be formatted after a minute or so, and once completed, a prompt asking for a disk label will appear. Either type in a label and press enter, or just press enter (the label is optional). When prompted to "Format another (Y/N)", type n and press enter.
- Copy all the files included in the converter package to the floppy disk. To conserve space on the floppy disk, the file `cmanual.pdf` can be omitted (see the converter file listing in an earlier section of this manual for specific information on files not required for normal operation of converter)
- Check the `autoexec.bat` file on the disk and edit it (using Windows' Notepad or Edit under DOS) in such a way that the "SET BLASTER" line reflects the specific settings of your soundcard (or if using a Gravis Ultrasound, replace the "SET BLASTER" line with the appropriate "SET ULTRASND" line for your card's configuration), and save the file.
  (For more information on determining the hardware settings to use for your particular card, see the previous appendix section.)
- If using a mouse or other pointing device for midi conversion, place its driver on this floppy and put the command to load it (ie. `mouse`) on the next line in the `autoexec.bat` file (before the line with '`c`', which loads converter – the mouse driver must be loaded before converter).

With this newly-created bootable disk, you just put it in your disk drive, boot up your computer (or reboot it), and converter automatically loads and runs without intervention. It also means converter can easily be run on other machines – all that is needed are (possibly) different numbers in the "SET BLASTER" or "SET ULTRASND" string in the `autoexec.bat` file if the other machine(s) have different configuration settings.

Note that this feature assumes your computer is configured to check the floppy drive for a disk when it is turned on – if it does not, check the manual that came with your motherboard or computer for a reference on the bootup BIOS settings which typically allow you to enable this feature.

tips to ensure best speed / performance

As with any complex system open to many different configurations and applications, there are certain factors which can influence converter's performance. While converter should provide satisfactory multi-channel audio to midi real time performance on any Pentium 100MHz or faster, there may be individuals attempting the use of multiple audio channels in combination with several other conversion input sources (or other complex applications for converter) on slower machines (such as a 486) looking for ways to squeeze some extra performance out of their configuration. As a summary of issues discussed in this manual which potentially affect the speed or response time of converter in live performance, the following tips and suggestions are designed as a general guideline for those wishing to improve performance on a slower machine.

- **Figure out what your particular machine's speed is capable of with converter**; avoid overloading the computer by trying to do too much at the same time. Specifically, if speed is an issue:
    - Try using the midi-only or audio-only core input modes instead of the simultaneous audio and midi input modes in order to provide a higher guarantee of reliability, lower response times, and lower processing overhead on slower machines
    - Disable the audio filters / filter channels if they are not needed – these take significant cpu power for slower (486, Pentium 60MHz) machines
    - Disable the gameport input (realtime settings menu) if it isn't being used – again, proper gameport interface routines take a consistently generous amount of cpu time, mostly in terms of a wait-state (due to the inherently cheap, ancient, and poor design of the IBM analog gameport interface). Even though the gameport routines within converter have been designed in the most efficient way possible (and they will not interfere with the timing of midi data), there is still a certain unavoidable load this interface type places on the host computer.
    - Disable the lfo generators (realtime settings menu) if they aren't being used. These *do* add an additional computing load, which although minimal, may be significant on slower machines.
    - Under more extreme situations, you might try setting the screen update option (root -> display settings) to minimal.
- **Do not use the vertical refresh interrupt option.**
- Use the 'Hardware & System Settings' or 'Audio and Filter Settings' display panel ([F11] or [shift]+[F11]) instead of the oscilloscopes display panel [F10] which requires much greater cpu time than the other display panels.
- Monitor the midi bandwidth generated by converter by checking the midi data rate meter in the hardware & system settings view screen (F11). Remember that midi's bandwidth (number of bytes it can transmit per second) is about 3000 bytes – typically keeping it under about 1500 or so is usually a good idea. **If converter's performance is less than desirable, it almost certainly has to do with the density of the midi output stream** (number of midi bytes which are required to be transmitted every second). Reducing the output midi stream density can be performed by making greater use of the audio data reduction algorithms if many channels of continuous controller audio to midi conversion are being used, reducing the number of LFOs being used (if several LFOs are programmed to generate their own midi data streams), disabling gameport input, and the like.
- Monitor the computer's processing load by checking the processor buffer peak meter at the upper left of the screen – it should never be 'maxing out' constantly (even though the buffer's actual 'max' is much higher than the peak meter's max position).
- As a last resort, disable screen updating and see if that improves the situation by entirely disabling the real-time graphics.

gameport interface pinout

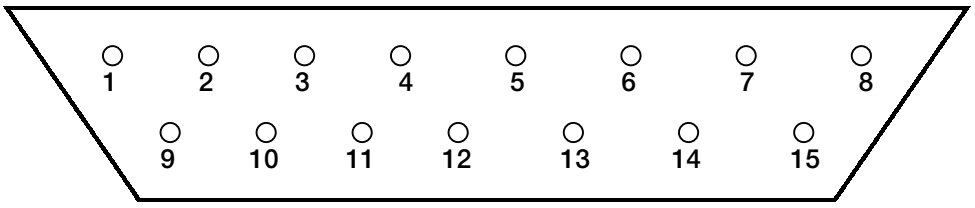
Warning! The following information is provided for users with concrete electronics knowledge and experience. There are 5-volt pins on the joystick interface that can provide a surprising amount of current if short circuited, causing potentially expensive and/or irreparable damage to components in the computer, as well as information loss, and any other number of disasters. By providing the following information, urr Sound Technologies Inc. in no way encourages general users of converter to attempt the construction of devices for interface to the PC gameport, and is in no way liable for any damages or loss of data as a result of the construction and/or use of any device based on the following information.

End of disclaimer ☺

Note that there are two differences on the joystick port found on a soundcard, such as the Gravis Ultrasound or SoundBlaster, versus the original IBM-spec PC gameport interface – pins 12 and 15 are assigned to midi functionality instead of the original ground and +5v designations.

Use 100kΩ - 150kΩ potentiometers on the axis inputs, preferably with a linear response curve.

joystick D-sub connector pin numbering



Original Gameport Pinout:

pin	connection
1	+5 volts DC
2	joystick A button 1
3	joystick A x-axis
4	ground
5	ground
6	joystick A y-axis
7	joystick A button 2
8	+5 volts DC
9	+5 volts DC
10	joystick B button 1
11	joystick B x-axis
12	ground
13	joystick B y-axis
14	joystick B button 2
15	+5 volts DC

Soundcard Gameport Pinout:

pin	connection
1	+5 volts DC
2	joystick A button 1
3	joystick A x-axis
4	ground
5	ground
6	joystick A y-axis
7	joystick A button 2
8	+5 volts DC
9	+5 volts DC
10	joystick B button 1
11	joystick B x-axis
12	MIDI TXD (Transmit)
13	joystick B y-axis
14	joystick B button 2
15	MIDI RXD (Receive)