WAVE-to for Windows

Contents

Technical Support

If you encounter any problems installing or running WAVE-to for Windows, you should first contact the suppliers of the software.

In the United Kingdom, this will be:

Optech Limited,
East Street,
Farnham,
Surrey.   GU9 7XX
Tel: +44 (0)252 714340
Fax: +44 (0)252 711121

Electronic Mail contact points:

Compuserve:
Kim Boulton   100114,2136

Internet:
kim@optech.demon.uk.co

Introduction to WAVE-to for Windows

WAVE-to for windows is a Visual Sample Editing and Conversion   package with support for MIDI Sample Dump it is intended to enable Synthesiser and Sampler users to utilise samples from a wide variety of sources.

An increasing number of modern synths offer user sample RAM built-in or as an option, including:

GEM S2 and S3
Korg T1, T2 T3
Kurzweil K2000/2000R
Peavey DPM3SE
Yamaha SY99/SY85/TG500

WAVE-to for Windows will convert to and from most common sample formats, including Windows 3.1 WAV files, Sample Vision (SMP), Sample Dump (SDS), Gravis UltraSound (.PAT), Amiga / Apple (AIFF), Amiga MOD, SAM and 8SVX, SoundBlaster (VOC) files, Wired for Sound and SoundTool files (SND), and a variety of Sun / Dec / UNIX / NeXT files.

Samples and   Loops may be auditioned on any PC provided that it is equipped with a device and driver capable of supporting Windows WAV files.

Used with a MIDI-equipped PC, samples can be dumped directly to a Synthesiser or Sampler using the MIDI Sample Dump protocol, or saved to disk as a .SDS file which can be interpreted by a Sequencer as a SYSEX message

This program will be of particular interest to owners of Yamaha SY85 synthesisers who don't have MIDI capabilities on their PC's, as it will read and write SY85 format waveform files.

WAVE-to employs full graphical editing facilities including cut, paste and zooming. It also allows automatic loop point location, cross-fading, and on-screen manipulation of sample loop points.

WAVE-to for Windows is Copyright Richard R. Goodwin 1993, 1994. All rights reserved.

Copyright and Disclaimer

WAVE-to for Windows is Copyright Richard R. Goodwin 1993, 1994. All rights reserved. Use, Duplication, or Sale of this product except as described in the License Agreement is strictly forbidden.

Users of WAVE-to for Windows must accept this disclaimer of warranty:

"WAVE-to for Windows is supplied as is. The Author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Author assumes no liability for damages, direct or consequential, which may arise from the use of WAVE-to for Windows".

File menu

The File menu gives access to the Waveform conversion 'filters' through the **F**ile **O**pen and **F**ile Save **A**s menu commands.

The file dialog box contains a set of 'Radio Buttons' for selecting the Sample file Type to be Read or Written.

A brief description of the file type is given in a panel at the bottom of the form. Clicking on the 'More' button will bring up the full description of the selected file format from the Help file.

Once a file has been opened, the **Save As** menu becomes available.

Whenever there is data in the clipboard, the File New menu is enabled.

Toolbar Equivalents

File New

File Open

File Save As

File New

Selecting **File New** clears the current display and brings up a blank waveform display into which data in the clipboard can be pasted.

This menu item is only enabled if there is valid data in the clipboard.

Toolbar Equivalent

🗋 File New

Edit Menu

The Edit menu is only available once a waveform has been opened and read-in for display.

The Edit Menu provides access to a standard range of editing functions including Cut, Copy, Paste and Crop.

These functions are fully documented in the Sample Editing topic.

Toolbar Equivalents

Edit Cut

Edit Copy

Edit Insert Paste

Edit Crop

Setup Menu

The Setup menu allows the user to select MIDI and Wave devices.

The **Wave Devices** screen also records the Sample Rate and Width that will be used when auditioning a sample.

Once installed correctly, WAVE-to will re-open the same ports as were specified the previous time it was run, except if the hardware configuration changes, in which case it will prompt the user to reselect which MIDI and Wave devices to use.

MIDI Menu

The MIDI menu provides access to the MIDI Sample Dump routine. This function is only available if supported by the PC hardware, and if a valid MIDI port has been opened.

Toolbar Equivalent

Zoom Menu

The Zoom Menu allows the user to magnify selected portions of the displayed waveform up to a maximum resolution of 20 displayed samples.

At resolutions of less than 200 displayed samples, each individual sample on the waveform is identified with a cross.

Toolbar Equivalents

🔍	Zoom Selected.
🔍	Zoom Whole Waveform

See Also

Wave Display
Sample Editing

Transform Menu

The following Transformations and Waveform Manipulation tools are available under the Transform Menu

ReSampling
Muting
Envelope Shaping
Loop Finding
Layering
Cross-Fade Loops

Toolbar Equivalents

Mute Selected

Envelope Shaper

Loopfinder General

Cross-fade looper

Play Menu

The **Play** menu provides access to functions for auditioning the sample and for generating MIDI note messages.

Auditioning

Four Auditioning mechanisms are provided:

Audition Loop

The portion of the waveform from (and including) the Loop Start up to (and including) the Loop End are repeated a pre-determined number of times.

The number of times that the loop is repeated is determined by the settings in the **Wave Devices** screen.

Audition Wave

The sample in memory is played in its entirety from start to finish.

Note that because PC soundcards play WAV data, and the internal data format used by WAVE-to is 32-Bit, WAVE-to has to convert a sample to WAV format before it can be played.

This will inevitably result in a delay before any sound is heard.

Audition Display

Just the displayed portion of the waveform will be played.

Audition Selected

Just the highlighted regon of the waveform will be played.

MIDI Note Generation

WAVE-to for Windows incorporates two different mechanisms for generating MIDI note data in order to trigger samples in an external synth / sequencer.

Velocity Plane

The Velocity Plane looks just like a blank window. Click anywhere within the window and a MIDI note on message is sent via the currently open MIDI Device on the currently selected MIDI Channel.

Moving horizontally the window is scaled to generate Notes from 0 (C-2) to 127 (G8).

Moving vertically the window is scaled to generate Note velocities from 0 (Note Off) to 127.

Clicking and dragging with the Mouse Pointer will generate a rapid-fire sequence of MIDI Note messages useful for trialling patches / sample assignments over the full range of Pitch / Velocity combinations.

Stylo-Key

The stylo-key is a simple MIDI keyboard playable using the Mouse. Its two octaves keyboard can be transposed to span the complete MIDI range of C-2 to G8.

The Note-on velocity can be changed over the full range of 0-127 using a slider.

Notes will be sent using the currently assigned MIDI Channel

Toolbar Equivalents

Audition Loop

Audition Waveform

Velocity Plane

Stylo-key

Converting Waveforms

Internally, WAVE-to for Windows treats all waveforms as a series of (4-Byte) signed long PCM samples. This gives WAVE-to for Windows a maximum resolution of 32-Bits.

The maximum editable waveform size is entirely dependant on the available memory in your PC, though its worth remembering that during some of the edit and manipulation functions, WAVE-to might actually need to create buffers up to 8 times the size of the waveform being read, which might mean that Windows runs out of memory before you expect.

Generally speaking, on a PC with 4Mb of RAM, samples up to 2Mb in size can be editted with ease.

Only Mono samples are supported, though where appropriate the user is given the choice as to whether to load the Left or Right channel.

Translation 'filters' are used to convert the samples being read from and written to disk.

Sample translation is achieved by using the "**F**ile **O**pen" and "**F**ile Save **A**s" menu functions.

The File dialog box contains a set of 'Radio Buttons' that determine the format of the files being Read and Written. A short description is given at the bottom of the dialog box. Clicking on the 'More' button will bring up the full help text for that particular file type.

The basic sample details (sample frequency, number of samples, resolution, loop points and type) are available under "**Sample Details**" in the File Menu.

When saving, additional dialog boxes will be displayed where appropriate to capture other information to be stored with the converted sample. For example, some formats support sound clip names, others support multiple loop points, markers etc.

Once read from disk, full sample viewing and editing facilities become available. Thereafter, the sample can either be re-saved to disk in the same format as it was read, or else it can be converted to any one of the other available formats.

Additionally, the sample can be Auditioned or dumped using MIDI Sample Dump Standard (MIDI-Equipped PC's only).

See Also
Universal Sample Reader
Sample Editing
Sample Layering
Sample Loop Points
Re-Sampling

Wave Display

A waveform will be displayed in its entirety immediately it has been loaded.

The horizontal and vertical scales are automatically adjusted for the sample resolution and number of samples in the waveform.

The status bar at the bottom of the screen displays details relating to the displayed portion of the waveform, including the Sample Numbers of the first and Last samples displayed.

The samples corresponding to the Loop Start and Loop End values are indicated by coloured vertical lines (see Sample Loop Points) Green for Loop Start, and Magenta for Loop End, and are also noted on the status bar.

Clicking and dragging with the left mouse-button will select an area of the waveform to zoom or edit.

Once a range of samples has been highlighted, the Zoom Selected, Edit Cut, Edit Copy and Edit Delete menu items and Tools become available.

Clicking on **Zoom Selected** will zoom the display to just that area that has been highlighted. Once zoomed, you can return to the full waveform display by selecting **Whole Waveform** from the Zoom menu.

Hitting **F2** or selecting **Zoom Region of Loop Start** will zoom in to the region around the Loop Start point, and will display 50 samples before and after the Loop Start point.

Similarly, hitting **F3** or selecting **Zoom Region of Loop End** will zoom in to the region around the loop end point.

The maximum zoom resolution is 10 samples. Trying to zoom beyond this resolution will have no effect.

At high resolutions (less than 200 displayed samples), the individual samples are indicated with a cross.

Colours are used merely to indicate a change in the sign of the waveform.

See Also

Sample Editing
Sample Loop Points
Converting Waveforms
Re-Sampling

Toolbar Equivalents

🔍	Zoom Selected.
🔍	Zoom Whole Waveform

Sample Editing

When a waveform has been loaded and is displayed, Sample Editing facilities are to be found under the **Edit** menu. Full sample editing facilities are available, including:

Undo	All destructive edits can be undone.
Cut	The highlighted region is removed from the displayed waveform to the Clipboard
Delete	Similar to Cut, except that the data isn't copied to the clipboard
Crop	Everything before and After the highlighted region is deleted
Copy	The Highlighted region is copied to the clipboard
Paste Over	Clipboard data is pasted on top of the existing data, starting at the start of the highlighted region
Paste Merge	Clipboard data is merged with the existing data, starting at the start of the highlighted region.
Insert Paste	Clipboard data is inserted into the displayed waveform at the start of the highlighted region.

Before Cut, Delete, Copy and Crop become available, you first have to mark an area of the waveform. Do this by holding down the left mouse button and dragging the cross-hair pointer across the screen.

Paste, Paste Merge and Insert Paste will take place at the start of a highlighted area.

All the standard Windows keystroke equivalents (Shift+Ins etc) also apply.

The RAW data when copied to the clipboard is not in a format that can be rendered or pasted into any other application, though I may look at translating it to RIFF format at some later stage to make it more widely available.

A number of functions are available for moving around the displayed waveform.

R-Arrow	Moves the displayed window forward by the a number of samples corresponding to one eighth of those currently displayed (if possible)
L-Arrow	Moves the displayed window backwards by the a number of samples corresponding to one eighth of those currently displayed (if possible)
+ (Plus)	Zooms the display by 50% (displays half as many samples) centred around the sample closest to the centre of the display, up to a maximum resolution of 10 samples.
- (Minus)	Zooms out by 100% (displays twice as many samples) centred on the sample closest to the centre of display.
F2	Zooms in to display (if possible) 50 samples above and below the current position of the Loop Start Marker
F3	Zooms in to display (if possible) 50 samples above and below the current position of the Loop End Marker.

F4	Displays the Loop Join
PgUp	Moves the displayed window forward by the a number of samples corresponding to one eighth of those currently displayed (if possible)
Shift+PgUp	Moves the displayed window forward by the number of samples corresponding to half of those currently displayed (if possible)
PgDn	Moves the displayed window backwards by the a number of samples corresponding to one eighth of those currently displayed (if possible)
Shift+PgDn	Moves the displayed window backwards by the number of samples corresponding to half of those currently displayed (if possible)
Zoom Selected	Zooms in to display just the highlighted region (up to a maximum resolution of 10 samples per page).
Zoom Whole	Zooms out to display the whole waveform.

At high resolutions (above 200 samples per screen), the individual sample positions are indicated with a cross. No attempt is made to artificially smooth the displayed waveform.

See Also
Sample Loop Points
Converting Waveforms
Re-Sampling

Toolbar Equivalents

Zoom Selected.

Zoom Whole Waveform

Edit Crop

Sample Loop Points

It's worth clarifying what WAVE-to understands as Loop Start and end points:

In WAVE-to, the sample buffer contains n samples, numbered from 1 to n. This is important to note, as some software will treat the first sample in a waveform as sample zero, and the last sample as n-1 (ie as an offset into a buffer of samples).

Wherever it is known, WAVE-to will adjust the Loop Start and End points to correspond to the sample naming convention of the format being written.

If this isn't known, you must be prepared to jog the loop points by one sample forward or backwards to get a clean loop.

By default, WAVE-to will set the Loop Start equal to the first sample (1), and the Loop End equal to the last sample (n).

When displaying a waveform, WAVE-to will place vertical 'markers' at the start and end loop points. These are incredibly useful when it comes to identifying those elusive clean loop points.

A Green marker represents the current loop Start point, and a Magenta marker represents the current loop End point.

Note that on most Windows screens I've tried, the far right-hand side of the screen is often not quite visible, which may mean that the Loop End marker is lurking just out of sight.

Loop start and end points can be moved either by typing the new values in the appropriate boxes on the Sample Details screen, or else by using the Right Mouse Button.

Clicking the right mouse button on the Sample Editing screen will move either the start or end marker (whichever is closest) to the indicated sample number.

Holding the right mouse pointer down and 'dragging' the mouse pointer will 'pull' that loop point around the display, though I'm afraid that this is a little slow in Visual Basic!

Hitting **F2** or selecting **Zoom Region of Loop Start** will zoom in to the region around the Loop Start point, and will display 50 samples before and after the Loop Start point.

Similarly, hitting **F3** or selecting **Zoom Region of Loop End** will zoom in to the region around the loop end point.

As a general principle, loops sound cleaner if the loop points are set at (or very close to) zero crossings.

Look at the waveform; Find a suitable point close to the position in time where you want the loop to occur, and set the loop start equal to either the sample just before or just after the waveform crosses the mid-point line. Do the same for the loop end, and try to avoid any rapid change in phase (direction), as it is this that is in the main responsible for those annoying clicks or glitches.

Finding clean loop points, particularly on a single instrument waveform is never easy, and takes a fair amount on intuition. Think of what the overall waveform would look like were you to take that portion of the waveform just before the Loop End, and tack it just in front of the Loop Start point.

You're trying to balance a number of factors including:

Amplitude	Abrupt changes in amplitude (volume) will click.

Phase	Make sure that the waveform doesn't change direction at the loop point
Frequency	Make sure the 'slope' of the waveform is the same before and after the loop point.

When you think you've got it right visually, it's always worth auditioning the loop, then jogging either the Loop Start or Loop End point forwards and backwards one or two samples, and comparing the difference.

Use the **F2** and **F3** keys to alternate between the loop start and loop end. Ideally, the waveforms should look identical, with the loop end marker one sample in advance of the equivalent position of the loop start marker.

See Also
Loopfinder General
Cross-Fade Looping
Sample Auditioning

Toolbar Equivalents

🔁	Audition Loop
🔊	Audition Waveform
FIND	Loopfinder General
🔍+	Zoom Selected.
🔍−	Zoom Whole Waveform

Sample Auditioning

WAVE-to will support Waveform and Loop auditioning on any PC equipped with a sound production mechanism with associated .WAV drivers. This includes virtually all PC sound cards from the most expensive hard-disk recording cards down to the humble PC speaker provided that a suitable .WAV driver has been loaded.

Because the internal data format used by WAVE-to is 32-Bit PCM, and the audio drivers on a Windows-based PC can normally only support .WAV file playback, WAVE-to has to convert the sample into .WAV format before it can play it. This inevitably results in a delay until the sound is actually heard.

Four auditioning mechanisms are provided:

Audition Wave	The waveform currently in memory is played in its entirety from start to finish as a single One-Shot.
Audition Loop	The samples from (and including) the Loop Start up to (and including) the Loop End are repeated a number of times.
Audition Selected	The currently selected region of the waveform is played once only.
Audition Displayed	Just the currently displayed portion of the waveform is played.

Within Windows, there are actually two mechanisms available for repeatedly playing a section of an audio waveform.

The simplest mechanism is to utilise a feature of the Windows audio drivers which allows you to specify the number of times to repeat a sample.

The second method involves repeatedly supplying buffers of audio data to the audio driver. The audio driver calls-back the program that sent it data to say that it is ready for more.

WAVE-to uses the former method rather than the latter, as Visual Basic is quite slow at responding to these call-back messages, which could result in glitches.

Note: Some .WAV drivers, notably those for the PC Speaker do not repeat a sample smoothly, but rather repeat it after a small (but very annoying) gap. This will result in nasty audible glitches of just the sort youve been trying to eliminate.

Toolbar Equivalents

🔁	Audition Loop
🔊	Audition Waveform

Loop Number

For reasons explained in the Sample Auditioning topic, WAVE-to only Loops a sample a certain number of times during playback.

The Number of Times to Loop field in Wave Devices setup screen allows you to change this value.

Whilst working on a sample with a very short loop period you might like to increase this number to let you listen to a decent length of the looped sound.

Similarly, listening to a 20 second sample repeated 500 times *can get a little boring.*

The value set will be saved in the configuration file and re-used in the next session.

Re-Sampling

The ReSampling function can be found under the **Transform** menu whenever a waveform is currently on display.

In a synth or sampler, where sample space is at a premium, a waveform can be resampled at a lower frequency to reduce the number of samples and thereby the overall size of the sample.

Some Synths / Samplers can only play samples recorded at one of a number of sampling rates. The common ones being:

8,000Hz
11,025Hz*
22,050Hz*
41,100Hz*
48,000Hz

* Standard Windows .WAV file sampling rates

When ReSampling to save space, keep the number of Bits Per Sample the same, and gradually reduce the sample frequency until a point just before the reduction in audio quality becomes unacceptable for the application you're using the sound in.

Some sounds can be re-sampled more successfully than others. A pure instrument tone will start to sound noisy quite quickly, whereas a Hi-Hat can be re-sampled to quite a low frequency.

Speech, also, can be re-sampled to a very low level and still remain intelligible.

The ReSampler in Wave-To can also change the sample resolution (number of Bits Per Sample). This is useful if, for example, you need to play 16-Bit waveforms on 8-Bit sound cards.

Another reason to change the sample resolution is to reduce the time taken to dump the sample using MIDI Sample Dump. The Sample Dump protocol is most efficient at transporting samples that have a resolution that is a multiple of 7-Bits (ie 7, 14, 21, 28).

An 8-Bit sample re-sampled to 7-Bits will take ***half*** the length of time to transmit using MIDI Sample Dump.

The down-side of reducing the sample resolution is that it will also reduce the dynamic range, and increase the noise.

Noise is introduced in the re-sampling process as the ReSampler approximates an original sample to the new available sample levels.

WAVE-to uses Linear Interpolation, and Floating-point arithmetic to determine the new sample value.

Having determined the position in time of the new sample, WAVE-to takes the samples immediately before and after the new sample location, draws a straight line between them, and picks the new value off that line.

Most audio waveforms are sampled using Linear PCM. That is to say that the total voltage of the waveform is broken down into a number of evenly-spaced discrete levels (256 for an 8-Bit, 65535 for a 16-Bit).

Other sampling formats, however, more accurately reflect the way we hear sounds.

The sound formats supported by Sun / DEC / UNIX / NeXT are often capable of supporting uLaw or aLaw non-linear PCM samples. These are standard companding techniques employed in telephony they normally employ a sampling frequency of 8,000Hz.

The .AU file reader will read an 8-Bit uLaw sample and expand it to a 16-Bit linear PCM sample for internal representation.

The .AU file write 'filter' is capable of companding a 16-Bit linear PCM sample to 8-Bit uLaw, but it will only do so if the sample is both 16-Bit, *and* sampled at 8,000Hz.

Microsoft have recognised the fact that the PC is now becoming a serious tool for the musician, and are attempting to widen the standards that apply to audio file encoding to incorporate advances in audio data compression and companding.

This has resulted in a number of open and proprietary standards being added to the Windows RIFF file definitions.

WAVE-to will read a number of these, though some of the proprietary formats are, in effect, closed standards, and are therefore not capable of being read by WAVE-to.

Mute Selected

Selecting Mute Selected from the Transform Menu will zero each sample in the selected range.

Note that this is a destructive process which cannot be undone.

Toolbar Equivalents

▣	Mute Selected

Envelope Shaping

A six-point envelope can be applied to any portion of the displayed waveform. Using the envelope shaper the amplitude of the waveform can be varied from zero to 200% over the displayed range of samples.

The amplitude of the samples between the points of the envelope is determined by linearly-interpolating the amplitude from the adjacent points.

The envelope is displayed as 6 points superimposed on the upper half of the displayed waveform. Initially they are all displayed in a line which represents the current amplitude (100%) each point can be dragged down to the Mid-point line (representing Zero amplitude) and up to the top of the displayed window (representing an amplification of 200%).

The envelope points can be dragged sideways as well as vertically, and by so-doing in conjunction with a suitably-zoomed display the amplitude of the resulting waveform may be very accurately trimmed.

Once you are happy with the appearance of the overlaid envelope, this can be applied to the displayed portion of the waveform by selecting **Envelope Apply** from the **Transform** menu.

Note that this is a destructive process that cannot be undone.

An Envelope Shaper is also employed in the UltraSound .PAT file write filter.

Toolbar Equivalents

Envelope Shaper

LoopFinder General

Identifying clean loop points is the Holy Grail of the Sample Creator. The LoopFinder is an attempt to aid this process through the mathematical comparison of a chosen loop point with the remainder of the waveform.

When first invoked, the Loopfiinder gives you the choice of whether to find a match for the current Start or the current End loop point, along with the number of samples to include in the search process, and the direction of the search (forwards towards the end of the waveform, or backwards towards the start of the waveform.

The fewer samples included, the more likely that a 100% match will be found, but this doesnt guarantee a good loop.

Including a greater number of samples (Say 1000) will normally result in a better loop.

The Loopfinder display is split in two. The top half displays the Samples included in the pattern the lower half displays the current best fit.

Each fit is given a mathematical percentage accuracy, and as better matches are found they replace the previous one.

A drop-down list box at the bottom of the screen stores all the matches that were found. It may be that a 97% fit towards the beginning of a sample actually sounds better than a 99% fit found later on.

This process is not fool-proof by any means, but is occasionally able to spot loops which otherwise wouldnt have been evident.

See Also

Cross-Fade Looping

Toolbar Equivalents

Loopfinder General

Cross-Fade Looping

Some samples just dont lend themselves to looping. For example percussive sounds might always click, as there can often never be a decent point in the sample to place the loop.

The mechanism for smoothing-out the loop point to reduce the impact of the change in the waveform is called **Cross-Fade Looping**.

There are a number of distinct types of cross-fade loops as detailed below, but all work in essentially the same way, which is to merge samples from the start of the loop with samples from the end, and thereby reducing the audible impact of the change.

The term Cross-Fading can also be applied to Sample Layering - the process whereby one sample is smoothly cross-faded into another.

The common types of Cross-Fade are:

Normal Cross-Fade
Reverse Cross-Fade
Bowtie Cross-Fade
Bi-Directional Cross-Fade

Toolbar Equivalents

Cross-fade looper

See Also

Sample Layering

Normal Cross-Fade

A normal cross-fade loop is generated by first identifying a suitable loop point in the sample and then merging samples from just before the loop start, with samples from just before the loop end.

This has the effect of smoothing-out any abrupt change in amplitude and phase as the sample loops, and help to reduce clicks and buzzes.

A simple cross-fade like this will work well with periodic waveforms that are fairly constant in tone.

Reverse Cross-Fade

A reverse cross-fade loop is generated by first identifying a suitable loop point in the sample, and then merging samples from just after the loop end, with samples from just after the loop start.

A reverse cross-fade helps to eliminate abrupt changes in tone, and will smooth out the tonal content of the loop.

This works best with multiple sound sources such as String Sections, Choirs and Brass Ensembles.

Bowtie Cross-Fade

Think of the Shape of a Bowtie! This is used to equalise the data around the loop point so that the transition in and out of the loop splice point is very smooth.

This works well on sounds with lots of ambience, reverb or other effects.

The Bowtie cross-fade estimates the average amplitude of the waveform before and after the loop point, and ramps the volumes such that they meet at a mid-point.

Bi-Directional Cross-Fade

A Bi-Directional Cross-Fade is optimised for use with bi-directional loops, especially useful for ensemble sounds such as a string section.

This created by applying both a forward and a reverse cross-fade to the loop point.

Sample Layering (Cross-Fading)

This is the process whereby one sample is smoothly cross-faded into a second.

In WAVE-to, the **CrossFade Selected** menu item becomes available when a region of the displayed waveform has been highlighted.

The volume of the first half of the selected region is ramped-down, whilst the volume of the second half is ramped-up, thereafter the first half is merged with the second half.

This will have the effect of shortening the sample by half the selected region.

MIDI Sample Dump

MIDI Sample Dump Protocol was incorporated into the MIDI specification to enable MIDI devices to exchange waveforms.

Sample Dumps are slow!! Tooth-grindingly Nail-bitingly slow!!

MIDI data is transmitted at a rate of approximately 3000 data bytes per second, at 8-Bit resolution, each sample will be represented by 2 MIDI data bytes, so taking a 5 Second Mono sample, sampled at 22.05Khz, this will be 110,250 samples which will be transmitted as 220,500 MIDI data Bytes (plus a few header and framing bytes).

220,500 MIDI data bytes will take approximately 73 seconds (ie 1 Minute 13 seconds!!) not something you want to have to do in mid performance!

Unfortunately, there's nothing you can do about this, other than use synths / samples that have disk drives and load from them.

A Sample Dump File (Normally having a .SDS or .SYX extension) will contain the MIDI Data Blocks exactly as they would be transmitted by a Synth / Sampler. A Sequencer programs that can handle Sysex files (Such as Cakewalk Pro for Windows) will merely read and transmit these verbatim.

WAVE-to for windows can both read, save and transmit any waveform file as a Sample Dump file.

The Data in a MIDI sample dump message is always transmitted as 7-Bit data, which means that an 8-Bit sample will take 2 SDS data bytes to transmit, as will a 14-Bit sample.

A 16-Bit sample will take 3 data bytes, as will a 21-Bit sample.

7, 14 and 21 bit samples result in the most efficient transfer of sample information by Sample Dump.

Re-sampling 16-Bit samples to 14-Bit will reduce the resulting sample dump files (and also the transmission time) by 33%. Note, however, that your Synth may not be able to handle these more esoteric sample widths.

Note that MIDI Sample Dump messages are one type of SYStem Exclusive (SYSEX) messages and, as the name implies, are exclusive to one particular synth / sampler manufacturer.

WAVE-to supports the creation and transmission of one of the most common formats, though some early samplers transmit and receive a different MIDI Sample Dumps in a different (incompatible) format.

Universal Sample File Reader

WAVE-to is able to interpret Sample files produced by practically any method (other than proprietary and un-recognised companding techniques) through the use of the **Universal Sample File Reader**.

Setting the file type as **"Universal"** will cause WAVE-to to prompt you for the sample format details for the sample to be read.

As a general rule, if the resulting waveform looks OK, it's probably been interpreted correctly. If you get the format wrong, the resulting waveform will look awful!

Using the Universal Sample file Reader, you can repeatedly 'guess' the format of the wave file to be read until it looks OK on the Sample Viewer. From that point the wave can be edited and saved as any other format.

The details you have to supply to the Universal File Reader are:

Data Width	This is the number of bytes for each sample. Note that this is different from the Sample Width, as a 7-Bit sample may actually be contained in a 16-Bit data word.
Sample Width	The number of Bits Per Sample. Note that this is different from the Data Width, though it must always be less than or equal to the Data Width.
Channels	Mono, Stereo or Quad. WAVE-to expects the data for each channel to be interleaved.
Sign/UnSigned	Signed Data has a mid-point of Zero, with the waveform extending positively and negatively around that point. Un-signed Data has a minimum point of Zero, and a mid-point of half the maximum sample amplitude.
Byte Order	Different machines physically write data to disk in different ways, and there is a major difference between the way that an Apple MAC writes data compared to an IBM PC. PC's write data in **Intel** or **Little Endian** format, that is to say that values of greater than 8-Bits are stored low-byte first. Apple, Amiga, Atari, NeXT and SUN are examples of machines that write data in **Motorola** or **Big Endian** format, that is to say that values of greater than 8-Bits are stored in high-byte, low-byte order (the opposite way round to PC's).
Header Length	The header length sets the number of bytes to be skipped before WAVE-to starts to read sample data. This can be used to skip the header on a file so that only the sample data is read.

WAVE-to will save the settings that you use so that if you tend to need to interpret files of a particular type, you don't need to keep re-setting the settings.

Windows 3.1 WAV Files

WAV is the native sound file format for Windows 3.1 audio files, and as such is becoming widely used on PC platforms. WAV is just one specific sub-type of the Windows 3.1 Resource Interchange File Format (RIFF).

RIFF is a tagged file structure upon which many file formats can be defined. File formats based on RIFF can be future-proofed as format changes can be ignored by existing applications.

The structure of a RIFF file is similar to the structure of an Electronic Arts IFF file. RIFF is not actually a file format itself (since it does not represent a specific kind of information).

The *EA IFF 85 Standard for Interchange Format Files* contains some useful background on the uses of a tagged file structure.

RIFF files are made up of 'Chunks' each chunk is preceded with an ID which defines the data representation or *form type*.

Windows Waveform Audio Format files have a RIFF form ID of 'WAVE'.

The generic RIFF file format is capable of supporting any number of open and proprietary audio file encoding, compression and companding formats.

Initially, the only open format was MicroSofts PCM format (a straightforward linear PCM encoding format).

More recently, a number of open and proprietary formats have been registered, most notably:

WAVE_FORMAT_ADPCM
WAVE_FORMAT_IBM_CVSD
WAVE_FORMAT_ALAW
WAVE_FORMAT_MULAW
WAVE_FORMAT_OKI_ADPCM
WAVE_FORMAT_DIGISTD
WAVE_FORMAT_DIGIFIX
WAVE_FORMAT_YAMAHA_ADPCM

WAVE-to for Windows will handle a number of these encoding and companding formats, though often the format is proprietary.

WAVE-to for windows currently handles all data internally as a time-series of signed 32-bit mono samples. When it encounters a Stereo WAV file, WAVE-to will give the user the option to read either the left or the right channel. Thereafter the waveform is treated as being Mono.

AIFF file format

AIFF files are used by the Apple IIc/IIgs and SGI, and on the Amiga.

AIFF (like RIFF) is based on the Electronic Arts Interchange File Format, a tagged file structure.

Different 'Chunks' within the file are tagged with a four-character code and a Chunk Size. This allows a file reader to skip unrecognised blocks (chunks) of information within a file, future-proofing the file structure to some extent.

Sampling information, such as the number of channels, the sampling rate, number of bits per sample etc. are stored in the 'COMM' chunk.

A 'SSND' chunk contains the sample data itself. WAVE-to for Windows assumes that all sample data is contained within a single SSND Chunk.

Chunks can appear in virtually any order in the file, so the file reader needs to be able to cope with that.

Other chunks will be ignored by the AIFF file read filter.

Identified Chunks include:

MARK	Markers (sometimes used for Loop Points)
INST	'Instrument' Chunks
APPL	'Application' Chunks - Often include the Name & Version of the application that created the file.

One COMM and one SSND chunk are written by the AIFF file write filter.

Gravis Ultrasound PAT File Format

The Gravis UltraSound card uses Wave-Table synthesis to produce sound output. This means that either sampled data from actual instruments or other synthesised digital audio is stored in RAM on the board for playback by the dedicated sound processor chip.

The card can have up to 32 active voices depending on the aggregate sample playback rate.

The 32 voices are independently controlled, and can be producing different sounds concurrently. The card output is stereo, and voices can be positioned within the stereo image.

Currently, each Advanced Gravis UltraSound Patch (.PAT) file will contain a single instrument, though the specification actually caters for multiple instruments in a file.

Each Instrument can consist of a number or different waveforms 'layered' on top of each other. For example, a piano could have two layers. The first might be the actual tone from the strings, whereas the second could be the mechanical noises arising from the hammer mechanisms moving.

The Sound Processor on the UltraSound can cope with both Signed and Unsigned data. A flag in the header record specifies the format that the patch is in.

WAVE-to will read both the current (type 110) and obsolete (type 100) PAT files.

When writing, WAVE-to will only write the newer (type 110) file format. An additional screen is displayed where the user can specify such information as the stereo balance, the Wave Name, and modify the envelope.

Amiga 8SVX File Format

8SVX format appears to be one particular type of annotated IFF file, and is used by some Amiga MOD file editors.

Like AIFF files, 8SVX files use a Tagged file structure, with the file being sub-divided into Blocks.

The main sample data is contained in the 'BODY' sub-chunk. Other sub-chunks identified include:

VHDR:	Contains sample format details.
ANNO:	24 Character file annotation
CHAN:	Channel information

WAVE-to uses the Amiga AIFF sample conversion filter to convert the body, which works for 8 Bit Mono samples. This read filter should also cope with formats up to 16-Bit Stereo, but I've never seen any, and so don't know their VHDR format.

Amiga MOD file format

WAVE-to for Windows is able to read Amiga 'MOD' files generated by a variety of programs, though it won't generate any itself. If you want to create MOD files, you should look at purchasing a dedicated MOD file editor.

MOD files are, however, a good source of sample data which can be used in other applications provided that you can extract them.

The process of extracting samples from a MOD file is generally called 'ripping'.

A number of concealment techniques have been employed over the years, particularly by games program manufacturers, to prevent people from stealing samples from their programs.

WAVE-to doesn't purport to be a MOD file 'ripper'. It is able to extract samples from MOD files conforming to 7 of the most common file formats. If you want to rip samples from games, you'll probably have to look elsewhere.

MOD or 'Module' files contain 4-channel music. The Commodore Amiga, for which the .MOD file format was developed, has four sound channels each having independent sample rates (no mixing occurs). Two of these channels are played on the left speaker, two on the right one.

A "song" contains information about which note to play at each moment. It does not contain the actual sound data. A "module" is a song with the sampled sound data concatenated. This sampled data is stored in "instruments". Each instrument is previously recorded and stored as a ".SAM" file.

A song contains multiple "patterns". Patterns are played back in sequence. Each pattern contains 64 "notes" for all four channels. These notes are played back sequentially as well. Each note contains information about which sample to play plus sundry additional information. Each note takes 4 bytes, a pattern thus takes 1024 bytes. At the normal tempo, about 8 notes are played per second.

There is no standard sample rate for the samples in the modules. Often, the sound data is sampled at a rate called C-3. This rate is 16,574 Hz on an Amiga PAL machine, or 16,727 Hz on an Amiga NTSC machine. The instrument must play a C-3 note of 911 Hz to be in tune. The sample rate C-2 (8,287 Hz for PAL) is apparently even more popular (the instrument must play a C-2, 456 Hz).

For percussion, the sample rate sometimes is A-3 (about 28 kHz). (The Amiga timers are connected to the video circuit. The origins of the .MOD file format lies here in Europe, so much of the timing is based on PAL.) The different sampling rates for PAL and NTSC versions of the Amiga can give tuning errors between samples. Software-based "fine tuning" tries to solve this as much as possible.

The original Amiga 'SoundTracker' song format from 1988 by Karsten Obars has been expanded by several competitive products, especially Noisetracker by Mahoney and Kaktus. This expansion process has resulted in a number of incompatible file formats.

The main recognised variants are "M.K." (as used by ProTracker 1.1B) "FLT4" and "FLT8" (as utilised by StarTrekker), and "M&K!" used by a number of different MOD file editors. Each of these formats are assumed to contain 31 samples per 'Module',

All unrecognised formats are assumed to contain 15 samples per 'Module', the original format for MOD files.

The problem with this approach is that the read 'filter' will try and interpret almost any binary file as a 15-Sample MOD file. This will result in very unpredictable results if the file you are opening isn't actually in MOD file format.

When reading a MOD file, WAVE-to for Windows will a display a list of the available samples. Using the mouse,   you can select (check) one or more samples to 'pull'.

WAVE-to will concatenate all selected samples into a single waveform.   If you want to select just one sample from a MOD file, you should check only the name of that sample.

The samples in a MOD file almost always start with 4 Null bytes, the SAM read 'filter' strips these off, though I felt that it was unwise to do so in the MOD sample ripper, as I have seen MOD files where the samples don't have these leading nulls.

If you want to get rid of them, just use the sample editing functions and delete them!

Amiga SAM file format

SAM is the RAW sample file format used by several of the MOD file editors for the Commodore Amiga for storing the individual samples that go to make up a MOD file.

SAM files appear to be only 8 bit, and contain no formatting information whatsoever.

SAM samples are preceded by 4 Null bytes, which WAVE-to for Windows strips off before reading the remainder of the data.

When Reading, WAVE-to assumes that SAM files are 8-Bit Mono, therefore the number of samples is determined by the length of the file.

Sample frequency isn't specified in a SAM file, so WAVE-to assumes a nominal 22,050KHz.

When writing, WAVE-to will dump the current sample data using the Amiga AIFF sample data format, though with no sample header information except for 4 Nulls which always seem to precede SAM files.

I read somewhere that these Nulls are used by some tracker programs for silence - they set the loop offsets to the first few bytes of the Sample file, and loop round them. I haven't been able to confirm this.

If the Sample currently in memory is of greater than 8-Bit resolution, WAVE-to will warn the user before proceeding.

If a Sample file of greater than 16-Bit resolution is written using the Amiga AIFF sample format (minus the Header), I'm not at all sure whether anything would be able to read it. Certainly WAVE-to wouldn't recognise it as either SAM or RAW format.

Wired for Sound / SoundTool SND file format

Quite a few packages use an extension of .SND for a sound file. WAVE-to for windows will attempt to read those conforming to a couple of the more standard formats, notably those employed by Wired for Sound and SoundTool.

Based on an original design by Aaron Wallace (DSound), SND files of this type were used by Martin Hepperle in his SoundTool program, and again later by Aaron Wallace in his "Wired for Sound".

Wired for Sound won't list a sound clip unless there is a title present, so WAVE-to generates a title.

As far as I am aware, these .SND files are only ever 8-Bit.

WAVE-to will inspect the header on any SND file, and will warn the user if it recognises one of the SunOS / DEC / NeXT audio file formats.

In that instance, you'll have to try opening the file using the .AU 'filter'

If you want to write a SND file of the SunOs / NeXT / DEC type, use the .AU file write 'filter' and change the file extension.

See Also
SunOS / DEC / NeXT .snd Files

SunOS /DEC / NeXT Audio file format (.AU)

Audio sound file format as used under SunOS 4.1 similar to the Audio format used by DEC, and on NeXT machines (where they are called SND files).

It can support a wide variety of formats including:

u-law
linear 8-bits
linear 16-bits
linear 24-bits
linear 32-bits
IEEE FP 32 bits
IEEE FP 64 bits

The files can contain multiple channels. WAVE-to will prompt the user to select either the left or the right channel.

WAVE-to for Windows will not support waveforms of greater than 32-Bit resolution.

WAVE-to will identify 8-Bit uLaw Samples, and convert them to 16-Bit linear.

uLaw is a non-linear sample encoding (companding) format used a lot in telephony. Basically, what would normally be thought of as a 16-Bit linear sample is converted to an 8-Bit using a Signed 4-Bit mantissa and a 3-Bit exponent.

This companding mechanism more realistically represents the way we perceive sounds, and ensures that the sample is encoded with a higher resolution close to zero.

By convention, uLaw samples are always 8000Hz, and WAVE-to uses this fact to determine whether to bring in uLaw compression when writing a Sun NeXT audio file.

When writing a Sun / NeXT file, ***if*** the file is 16-Bit and sampled at 8000Hz, WAVE-to will prompt the user whether it should convert the wave to 8-Bit uLaw.

See Also
The Wired for Sound / SoundTool file reader.

RAW Data Files

RAW data files in this context, are files of Sample Data as held internally by WAVE-to, if you need to interpret a sample format that isn't currently supported, then you should use the Universal Sample File Reader which is capable of reading virtually any Sample File format.

RAW data files are, as their name implies, the RAW sample data. Internally, WAVE-to for Windows treats all waveforms as a time-series of Signed 32-Bit data words (Longs).

RAW data files don't contain any sample format information - this has to be determined by inspection.

RAW data files are pretty inefficient on their use of storage, so will tend to be much larger than waveforms in other formats. They do have the advantage however that they can be manipulated mathematically, as they are in the form generally expected by Mathematics packages.

When reading a RAW data file, WAVE-to for windows is able to determine the number of samples, the maximum sample width (sample resolution), but is not able to determine sample frequency.

If no sample frequency were present, many of the sample write 'filters' would error, so a nominal sample rate of 22.050KHz is assumed, even though 'Unknown' is displayed. The effect of this isn't critical, all that will generally happen is that the resultant sample will appear to be in a different original key than the one you expected.

Sound Blaster VOC files

VOC files are created and used by the Creative Labs Sound Blaster series of Audio Cards.

VOC files are quite complex, and can contain silent parts, loops, text, markers and different sample rates in different sections.

Originally, the file format supported only 8-Bit Mono samples, but Creative Labs added new sound block types when they brought out the more advanced SoundBlaster cards (which support 16-Bit stereo samples).

The Old-Style (Type 1), VOC files contained a single byte which approximated the original sample frequency, so the frequency of files converted from the Old format may differ slightly from the original.

The New-Style (Type 9) sound block header now includes the sample frequency in full, the number of channels, and sundry other information.

WAVE-to will skip unrecognised blocks in a VOC file, but will only read and convert the first sound block that it finds.

When Writing a VOC file, WAVE-to for Windows always uses the New VOC file sound block header format (Type 9).

Yamaha SY85 waveform files

The Yamaha SY85 music workstation can save and read waveform files from its internal disk drive.

This mechanism is preferable to using MIDI Sample Dump which is very slow.

It's possible to increased the internal User Sample RAM up to (theoretically) around 32Mb using industry standard SIMMS.

The internal disk drive on the SY85 is only 720K. When saving samples from the SY85, it will prompt you to enter a second disk when the first becomes full.

WAVE-to will currently only read and write SY85 wave files that will fit on a single floppy disk (ie have a maximum size of 720K). If you need to transfer larger files to an SY85 you'll have to use Midi Sample Dump protocol (and wander off and do other things for a while!).

SY85 waveform files have a .Wxx extension (eg DING.W01), where 01 is the 'Disk' number in Yamaha parlance.

SY85 .Wxx files are always stored as 16-Bit data, so interestingly an 8-Bit sample will take up the same amount of disk space as a 16-Bit sample (with the same number of samples). This means that you can't save space by re-sampling to a lower resolution, though lowering the sample rate will make a difference.

I'm not sure whether this 'feature' is carried over into the SY's internal RAM, but it's quite possible.

Turtle Beach Sample Vision files

Sample Vision by Turtle Beach is a software package that supports communication with various Samplers.

Sample Vision files are (I believe) always stored as 16-Bit mono, but obviously the samples themselves can be of any resolution up to 16-Bits.

Sample resolution must be determined by inspection, it's not stored in the file anywhere. The Read filter notes the maximum sample value encountered during the conversion process, which is then used to determine the resolution of the sample.

The file header has space for comments and a sample name.

A trailer at the end of the file stores the Loop, Marker and Sample Rate details.

A Sample Vision file can contain up to 8 Loops, each having a start and end sample number, a loop type (Loop Off, Forward and Alternate), and a Loop count (number of times to loop).

Additionally, the Trailer holds a SMPTE offset (in sub-frames) and a Cycle-Size (the number of samples in one cycle of the sampled sound), though these are optional fields.

Preferences

WAVE-to for Windows automatically saves the last Drive/Directory you used for loading and saving each file format. This cuts down the amount of directory browsing you have to do if you generally keep all your .WAV files in one directory, all your .SDS files in another Etc.

WAVE-to for Windows will retain MIDI Channel and MIDI Device assignments between sessions.

These preferences are stored in a .INI file (WAVE2.INI) which is created in your \WINDOWS directory.

MIDI Devices

On start-up, WAVE-to will interrogate the available MIDI Output devices in your PC and check their capabilities.

The program will warn you if none are available or if none are selected.

If no MIDI Devices are available or selected, none of the functions that rely on MIDI Data communication will be available, though all other functions will operate normally.

Only one MIDI device or port can be selected at any one time. Changing the selected device or port will close the one previously selected before opening the newly-selected port.

When you exit the program, your current settings are saved. If the configuration of your system remains the same, then the next time that the program is run, the same selected port will be automatically opened as the program starts.

If WAVE-to is unable to open the selected MIDI port, it will tell you. This normally occurs if you have another MIDI program (Sequencer Etc) already running which has the port held open, and the relevant Windows MIDI Driver will not allow itself to be shared by more than one program.

Some drivers will allow sharing, but unfortunately many will not (including the MIDI Mapper).

One way round the problem is to use a second MIDI port (for example two windows programs can use different ports on the same multi-port MIDI adapter), and to use an external MIDI Merge unit, though these are not cheap (they start at around £80).

I have heard of a windows utility which will allow several MIDI programs to share one driver, but I haven't as yet been able to find and/or test it.

MIDI Channel

Use this to select the MIDI Channel that you wish to communicate with you Synth / Sampler on.

The Synth / Sampler must also be set to expect data on the same MIDI Channel (or All / Omni).

This is useful in complex set-ups where different Synths / Samplers are chained together (ie addressed through the same port). They can each be set to different MIDI Channels in order to differentiate between them.

The MIDI channel setting is saved as you exit WAVE-to, and will automatically be used the next time that the program is run.

About the Author

Richard Goodwin is a Musician and programmer living in Kent, England.

I trained as a classical flautist from the age of 8, and have always had a fascination for electronics and the early synthesiser pioneers such as Leon Theremin, Bob Moog and Tim Orr.

I studied Electronics at the University of Kent at Canterbury, building a wind controller for my kit-built Transcendent 2000 monophonic synthesiser as my final year project using flute-like fingering and breath control. It was some years before I realised that this was quite advanced for 1981!!

I've spent 10 years working for City financial institutions designing and implementing information systems, and am currently Head of Information Technology at a Financial Information publisher.

In my spare time I play Saxophones, Bagpipes and a variety of more unusual woodwind instruments (Taragot, Chaleaumeau, Klarnet and Half-Long Bagpipes) in the Folk Band "Florida".

Florida are well-known in Folk Dance and Ceilidh circles in England, and have been regularly booked at most of the top 'dancers' festivals over the past couple of years. Their line-up comprises Mellodeon, Bass Guitar, Electric Guitar/Fiddle, Trumpets & Woodwind (yes we have no synths!).

Florida's 'almost traditional' style of music doesn't lend itself to the use of synthesised instruments, thus so far the synthesisers haven't made an appearance, though if things go according to plan, the Yamaha WX7 wind synth will soon start making an appearance.

Florida have released their first CD 'Splitting the Night' in 1992, with a second scheduled for release in Summer 1994.

With the advent of cheaper PC-based sequencers and MIDI, I got back into Synthesised music, and now have a Yamaha PF85 piano, a Yamaha TG500 tone generator, a Yamaha WX7 wind synth, a Korg M1, a Gravis Ultrasound, a variety of outboard effects and small 4-track.

The music I'm making is mainly backing tracks for my wind instruments, and arrangements for the band. I've got a couple of 'Multi-Setups' on the TG that replicate the lineup of the band which makes it pretty easy to bang out tune arrangements and play them to the rest of the group, two of whom don't read music.

When Microsoft brought out Windows 3.1 I picked up a copy of the Multimedia Programmers Reference, and realised that writing MIDI-aware applications had suddenly become a whole lot easier.

Then with Visual Basic came the ability to generate fairly slick front-ends so I set about writing an Editor for the TG500 which I'd failed to get anything useful out of by programming via the front panel switches.

Unfortunately I soon ran out of steam using Visual Basic alone, and had to resort to writing most of the processing-intensive routines in 'C'.

As part of the TG500 Editor, I wanted the ability to download readily available Windows Wave (.WAV) files to the TG in Sample Dump format, and to be able to accurately spot sample loop points. Having done that, I continued to develop the program into a more generalised sample editing / conversion program that I hoped might have a wider appeal, and removed anything specific to the TG, though the legacy remains in the name of the DLL

See Also
Contacting the Author

Contacting the Author

For technical support of WAVE-to you should first contact the supplier from whom you purchased the product (See Technical Support).

For all other general correspondence, I can be contacted via CompuServe, address CIS:100330,2442 [INTERNET:100330.2442@compuserve.com]

I cannot guarantee to respond within a given time, but I normally check CompuServe every couple of days.

I am always interested in any feedback you may be able to give me about the program.... Is it useful? Could it be better? What features do you use it for? Are there any other Sound File formats you would like to see supported? Have you encountered any problems using WAVE-to ?

Alternatively I can be contacted by post at:

Clare Cottage,
106 Clare Lane,
East Malling.
KENT   ME19 6JB
England.

Acknowledgements

Thanks to Andy Kuefmann CIS:100117,2226 who kindly gave me the source code for his WAV2SDS program which in turn gave me a good understanding of the PCM format of Sample Dump Standard files.

I am indebted to SY85 owner (and Trumpet player!) Donnie Guedry CIS:70324,2505 for helping me determine the SY85 file formats.

Thanks to the numerous Compuserve users who over the years have uploaded sound files in a variety of formats which has given me a wide range of source data to work on, and those who registered the Shareware version of Wave-to or took the time to provide me with feedback and bug reports.

I have used a DLL called POINTERS.DLL which I obtained from the Visual Basic forum on Compuserve. This freeware program was written by Greg Blaum CIS:71212,1763 and has proved invaluable in that it returns a Long Pointer to any Visual Basic object.

I haven't used any Sample Translation code from SOX7 (the sound Exchange), though Lance Norskog and Guido Van Rossum's 'C' source code clarified my understanding of the structure of both 8-Bit VOC files and SunOS .AU files.

I have, however, used Malcolm Slaney and Ken Turkowski's routines (included with SOX) to convert to and from extended precision IEEE floating-point format (as used by the Amiga for storing Sampling Rates), and also Craig Reese (IDA/Supercomputing Research Centre) and Joe Campbell's (Department of Defense) routines (also included with SOX) to convert to and from 8-Bit uLaw.

I obtained a description of the Amiga MOD file format from Thiadmer Riemersma CIS:100115,2074 who had compiled it from various (unacknowledged) sources.

Thanks to Jim Woodcock CIS:100064,513 for loan of his Compuserve account (I finally got round to getting my own!), for testing my install programs, and ultimately for lending me his SY85 for the weekend so that I could finally determine the unknown fields in the SY85 wave file header.

Thanks to Kim Boulton of Optech Limited for supplying a test environment, general feedback, and for shrink-wrapping WAVE-to.

Windows and Visual Basic are trademarks of the Microsoft Corporation.
CompuServe is a registered trademark of Compuserve, Inc.

Last but not least, thanks to Trisha and baby Stefan (b. 05/Jun/1993) for putting up with my failed attempts to communicate with my TG500.

SYSEX Messages

SYSEX or **SYS**tem **EX**clusive messages are, as their name suggests specific to a particular MIDI device.

SYSEX messages are typically used to convey machine-specific information from Synth to Sequencer or from Synth to Synth.

The only rules governing SYSEX messages from the point of view of the MIDI specification is that they must start with F0hex (SOX - **S**tart **O**f e**X**clusive), end with F7hex (EOX - **E**nd **O**f e**X**clusive) and only comprise 7-Bit data.

The general format is:

F0 [manufacturers id] [data] F7

The MIDI Manufacturers Association allocate specific SYSEX ID's in order to further separate the SYSEX messages of one manufacturer from those of another.

This leaves MIDI equipment manufacturers free to create a wide variety of machine-specific MIDI data messages that can easily be ignored by machines for which they are not intended.

MIDI Sample Dump messages are just one particular sub-type of SYSEX message, and are unfortunately not universally standard. Although more modern Synths / Samplers tend to conform to the protocol supported by WAVE-to, others, and particularly some early Sample Dump-equipped Samplers utilise SYSEX messages of a quite different (and therefore incompatible) format.

Glossary

Sound File / Sample Formats

8SVX
AIFF
.AU
MOD
RIFF
PAT
PCM
RAW
SAM
SIT
SND
SMP
VOC
WAV

8SVX: Amiga 8SVX files - Appear to be an annotated Multimedia file similar to AIFF

ADPCM: Adaptive Differential Pulse Code Modulation

AIFF: Multimedia archive file, used by Amiga, similar to EA-IFF-85. Also used on Apple II.

AU: sound file format used by SunOS 4.1 and NeXT.

MIDI: Musical Instrument Digital Interface

MOD: Amiga 'Module' files - A song file with concatenated samples.

PAT: Advanced Gravis UltraSound Patch File format - Multiple Waveforms can be layered to form a single voice for playback using the UltraSound's on-board 32-voice wavetable synthesiser.

PCM: Pulse Code Modulation

RAW Data: RAW data files consist of just the sample data in the same form as it is stored internally (ie as 32-Bit Signed Long datawords).

RIFF: Resource Interchange File Format. General Tagged file structure used by Windows and OS/2, Similar but not identical to IFF.

SAM: Amiga Raw Sample File format.

SIT: Files with a SIT extension are usually MAC files, compressed using 'STUFFIT'. These can be de-compressed in a PC environment using various shareware utilities such as UNSTUFIT or UNSIT. Sit Files include Resource and Data information - if you're un-stuffing an Audio file (Eg an AIFF file), un-stuff only the data portion of the SIT file.

SND: Wired for Sound / SoundTool sound clip files.

SMP: Turtle Beach SampleVision Sample file format.

VOC: Creative Labs Soundblaster Audio file format. Both the Old (8-Bit Mono) and the New (16-Bit Stereo) formats are supported (Types 1 and 9).

WAV: Waveform AudioFile - widely used by Windows & OS/2, it's a subset of the Windows Resource Interchange File Format (RIFF), which is similar to IFF. The .WAV file extension is becoming ever-more generic, and now covers a variety of different encoding and companding formats, though these all conform to the generall RIFF file format.

Taragot: An Eastern-European fore-runner of the Saxophone (Adolphe Sax didn't invent the Saxophone, he merely 'borrowed' the idea). The Taragot has a conical bore, a single reed, and over-blows an octave just like a Sax. Mine is keyed in 'D', and is made by Breton Bombarde makers Hervieux & Glet (contact me if you want more details). There was an English instrument called a Tarogato, but it died out quite a long time ago.

Chaleaumeau: A single-octave simple Clarinet. Clarinet's were all called Chaleaumeau's until some enterprising soul discovered the 'Clarinet register' (12 tones above the tonic), and turned the mouthpiece the other way up! Mine (I've three) are made by Breton Bombarde makers Hervieux & Glet (contact me if you want more details).

Klarnet: A Turkish metal Clarinet. Mine's pitched in 'D', and has a lovely low 'woody' sound. I got it from a little music shop up by the Basmane train station in Izmir (the ancient Smyrna) in Turkey.

Half-Long Bagpipes: are a Scottish lowland bagpipe, not to be confused with Scottish Highland pipes. Half-longs are a 'Cauld Wind' pipe, that is to say they are bellows blown like the Irish Uillean (Elbow) and Northumbrian pipes. Half-longs are quieter and more sonorous than Highland, but louder than Northumbrians. Mine are pitched in 'D', and are made by Jon Swayne (of Blowzabella fame).

Ceilidh: (Pronounced Kay-Lee) is the term normally used to describe English social dancing, the term deriving from the gaelic word for a social gathering. Ceilidh's are (I'm type-casting here) more fun than Barn-dances, which have a bit of a yee-ha image. Ceilidh's are the folk music equivalent of the Rave - they can go on all night, the music is normally pretty loud and fairly electric.

Change History

Version 1.0 20/Dec/1993

First cut, includes WAV and SDS filters, wave display, Midi download

Version 1.1 10/Jan/1994

Initial test release

Version 1.2 20/Jan/1994

Pre-release version, for testing Get and Put SY85 filters. Name changed from WAV2SDS to 'WAVE-to'.

First cut of Help file written.

Fields added to WAVE2.DAT to record the full path used for saving and reading each type of sample file.

Re-worked to operate using File Open.... File Save As.... functions, with my own file dialog box.

About (more) Buttons & information lines added to file dialog box.

Sub-menu's given Cancel and minimise buttons.

Changed Main screen bit map to curved Piano keyboard.

****** Released as Version 1.3 23/Jan/94 *****

GetSMP() written.

Bug Fix: GetWAV eliminated problem with wave not being displayed.

Delay added before DrawWave to enable WaveViewer window to be fully displayed before wave drawing routine kicks in.

Started on PutSMP(). Fields added to WAVE2.DAT to record the type of the last file read and saved. OpenFile and FileSaveAs routines changed to select the same type as that last used.

PutSMP() routine completed, though only tested for 16-Bit mono samples.

Outstanding problem with the routine that generates the trailing bytes of the file - these are generated by VB, and are subject to 1-Bit rounding errors when splitting a Long into a series of Bytes.

File Name (full path and filename) added to banner of Sample Details screen.

Tidied up the Global Memory allocation / reallocation to ensure that all locked global memory is freed when the program exits, and that no global memory is allocated or locked unnecessarily.

Forced each read filter to calculate the size of the resultant WAV buffer - avoids relying on keeping that information in a Global Variable.

Error routine to trap 'media not available' errors which occur (particularly on a laptop !!) if the saved drive specification is unavailable the next time that the program is run. In this instance, it now defaults to the application root directory (which certainly ought to be available, or how else would you be running the program?!!)

Program now detects whether the version number has changed (ie it's been upgraded but the Config File hasn't changed in length) and confirms to the user that the new version is running, but that the old settings have been retained.

****** Released as Version 1.4 02/Feb/94 ******

Finally decided that persevering with using WAV as the internal data format is un-workable.

WAV files use unsigned data for samples above 8 bit resolution, and signed data for samples below 8-bits. This means that all translation and display routines have to handle 8-Bit data separately.

Started work on converting ALL routines to work off a series of Signed Long (32-Bit) data words. This will give me a lot of flexibility, and simplify a lot of the sample manipulation routines.

Wav2Raw, SDS2Raw, SY852Raw, SMP2Raw and Raw Save filter completed.

GetRaw() Read filter completed.

Code to detect and choose left / right channels for Stereo WAV files.

Raw2Wav filter completed for 8 and 16 Bit WAV files. Program starting to become useful again after a week or so of being unuseable whilst I converted to a signed long internal data format.

Disabled all incomplete menu options.

Raw2SY85 filter completed for 8 and 16 Bit data.

******Released as Version 1.5 07/Feb/94******

Added better memory allocation error checking to all GetXXX() functions.
Disabled Save menu function, as Save isn't valid unless file has been altered.

Bug fix: Got rid of error message closing uninitialised midi ports if User selects "No" to no valid midi ports message on startup.

Created GetVOC() SoundBlaster VOC file reader for 8-Bit VOC files

Bug-fix: Cancel button wasn't being cleared - if you used it once it would remain pressed - all subsequent conversions therefore halted after the first packet.

Ported the routines to read an Amiga IEEE floating point value and convert it to a Visual Basic Double from 'C' to VB/C. Original C routines developed by Malcolm Slaney and Ken Turkowski, and included with SOX7.

Changed the install program to detect whether the user already has the VB runtime files, and prompt for whether they wan them to be overwritten.

Wrote the Sample Editing routines. Changed the Global memory allocation to use GlobalReAlloc() on minimal buffers created during program start-up.

Removed unused menus, including Timestretch and Play - they were frustrating people who used the program.

******Released as Version 1.6 10/Feb/94******

Debugged the sample editing routines. And re-checked all the ReAlloc routines.

Re-wrote the Midi Out handler to make it a little more robust.

Finished the AIFF file reader (only tested for 8-Bit Mono data, though should work with stereo).

Re-wrote the VOC file reader to cope with both the Old and the New VOC file formats, including 16-Bit Stereo. This version sent to Jim Woodcock as 1.6a.

Changed the Drag-Drop Icon on the Stylo-Key board to have true magic inverse.

Wrote VOC file writer, works for 8 and 16 Bit Mono files, and uses the NEW VOC file header format, could easily be modified to use the old style (or to give the user the choice).

Re-wrote the Sample Dump routines in 'C', as they were taking forever in Visual Basic

Started the AIFF file write filter. Currently don't have a format for 16-Bit AIFF samples.

Wrote the Double2IEEEFloat( ) routine to generate an IEEE floating point number from a Visual Basic double.

Worked-out the format for 16-Bit AIFF samples, finished the AIFF file read and write 'Filters' for both 8 and 16-Bit AIFF files.

Wrote the File Save routines.

Bug Fix: Changed the File Type Radio Buttons to also generate a Drive change event - this wasn't displaying the drive correctly in the Drive List box. Also works if drives become unavailable (eg. when a laptop is taken out of a base station).

Updated ReadMe and Help files. Version now ready for Release to Compuserve.

V1.7 sent to Yamaha, 22/Feb/1994.

******Released as Version 1.7 22/Feb/1994******

Bug Fix: Setup program needed SETUPKIT.DLL, which wasn't included on the distribution disk. Also, if the user didn't have THREED.VBX or CMDIALOG.VBX, it wasn't copying them over (though it was if they already had them!).

Bug Fix: Divide by Zero error if Old-Style VOC was first File format to be read.

******Released as Version 1.7a 26/Feb/1994******

Bug Fix: Didn't skip unrecognised blocks in Amiga AIFF files.

Wrote 8SVX file read 'filter' for 8-Bit files, but currently unable to determine sampling frequency, or number of channels, so am assuming they're always 8 Bit, 22050 Khz Mono

Wrote the Amiga SAM file reader - reads 8-Bit Raw SAM files - No header info or anything.

Wrote the Amiga SAM file write 'filter' This essentially dumps the sample data using the Amiga AIFF Ssnd Chunk format, though with no header. I haven't seen anything other than an 8-Bit SAM file, and there's no formatting information in the file, so the SAM reader won't be able to make sense of anything other than an 8-Bit file.

Warning screen warns users of this.

Bug Fix: Removed the problem with double slashes '\\' when reading or saving in the root directory of a drive.

******Released as 1.7b 01/Mar/1994******

Bug Fix: In the sample display, it wouldn't let you select the last few samples. This was because in VB, the X co-ordinate never actually equals the logical screen width, it gets within 99% of it, but not exactly to it.

Checked all the cut / paste and display routines to make sure that they were behaving themselves. In particular confirmed that samples always start at 1

Bug Fix: Re-wrote the Sample Display routine in the DLL to display the first sample exactly on the left hand side of the screen, and the last (selected) sample exactly on the right hand side of the screen. This removed a lot of ambiguity about which sample you were pointing at with the mouse.

Bug Fix: Clarified the situation in a 'Paste Over' command, where the pasted region overlaps the end of the buffer thereby increasing the length of the buffer. The number of displayed samples remains the same, but the **Zoom Whole Waveform** menu is now enabled. Total Samples and Samples Displayed accurately reflect the appearance of the buffer.

Added routines to draw markers at the current loop start and end points. Right mouse button can now be used to pick up a loop point and move it around.

Clicking with the Right mouse button will move the nearest loop point to the indicated sample.

Altered the VOC file reader to skip unrecognised blocks in VOC files.

Wrote the MOD file 'ripper' to extract samples from Amiga MOD files conforming to 7 of the most common formats.

Bug Fix: Removed the routine that cleared the edit and sample view menu's when a new file is opened - this means that if you cancel a file read, you can still go back and edit whatever was in the buffer to start with.

Added functions to zoom into just the regions surrounding the loop start and loop end points.

Added PageUp, Shift PageUp, PageDown, Shift PageDown, and Left/Right arrow keys to move around the displayed sample.

******Released as 1.7c 05/Mar/1994******

Added '+' and '-' keys to zoom in and out around the sample closest to the centre of the display.

Wrote the re-sampler, using linear interpolation. This threw up a bug in RAW data format for samples of greater than 16-Bit resolution.

Bug Fix: Found a far-reaching problem in the internal data representation (RAW) for samples of greater than 16-bit resolution. This has involved re-writing sections in just about every 'filter'.

Wrote the .SND file read 'filter' to read Wired for Sound and SoundTool 8-Bit file formats, and ammended it to recognise SunOS / DEC / NeXT file headers (it will warn the user and suggest they use the .AU file read 'filter').

Managed to track down a source of NeXT .snd files, so am now able to complete the .AU file reader for 8 and 16-bit mono files.

Bug Fix: Program used to crash windows if you selected 'Register' from the shareware registration reminder screen displayed when program exits (it doesn't now).

Bug Fix: Finally managed to borrow an SY85 (from Jim Woodcock), and so tracked down the bug which caused the truncation of long SY85 samples.

Bug Fix: Saving files to the root directory of the floppy drive still included an unnecessary second slash.

Completed .AU file reader for Sun / NeXT '.snd' files, including handling 8-Bit uLaw expansion to 16-Bit.

******Released as V1.8 15/Mar/94******

Wrote the Advanced Gravis Ultrasound patch (.PAT) file read and write 'filters', copes with signed and unsigned samples, and on saving, allows the user to modify sundry information.

Bug Fix: Modifying Loop points by typing the new values in the boxes on the Sample Details screen tended to reset the loop points to the beginning and end of the sample.

Started writing the Universal Sample File Reader, that will interpret ***any*** sample file formats - user needs to supply quite a few details.

Started adding Envelope shaping to PAT file writer and (tentitavely) to the Wave Viewer screen.

Bug Fix: Discovered that MAC AIFF files can have their COMM & SSND chunks in any order, so re-wrote the AIFF file reader to cope with that.

Bug Fix: 16-Bit AIFF file reader and writer appears to be wrong - Not sure where I originally determined the format from, but I now have a reliable source of MAC AIFF files, so re-wrote the 16-bit routines to read and write that format.

Completed the universal sample reader for all 8 and 16-bit waveforms plus a few other generic formats.

Bug Fix: Finally tracked down the reason why the Useage Count on the DLL (TG.DLL) kept climbing.

Bug Fix: Wasn't reading 12-Bit SY85 'WaveDisk' files correctly. It should now cope with anything from 8 to 16 bits in width.

Completed the Envelope shaper - Graphically applies an envelope to any portion of the displayed waveform.

Bug Fix: Now saves the base octave of the Stylo-key

Converted the program to Multiple Document Interface, which will eventually have a payback when it comes to editing stereo files and editing several files at once.

Added a startup screen to display the main Logo, this is no longer displayed once the program is fully up and running.

Started adding a toolbar to the Main screen.

Bug Fix: It was incorrectly subtracting 1 from the Loop Start and End on .PAT files, resulting in a -1 error.

Added status bar to bottom of main screen as somewhere to display the current displayed sample range, selected range - that sort of thing

Added Window Cascade and Tile commands to all menus.

Changed the re-sampler to re-display the current waveform after re-sampling has taken place.

Created the LoopFinder General.

Added waveform and loop auditioning.

Bug Fix: An updated version of THREED.VBX fixed a number of problems with 3D controls hanging onto the focus.

Completed the ToolBar

Added Mute Selected function to silence a selected portion of the waveform.

Added Reverse Selected to revers the sample order of a selected portion of the waveform.

Added ToolTips - These are displayed after a second or so if the mouse pointer is left over an enabled Tool.

Bug Fix: LoopFinder pattern-matching screen wasnt redrawing if it was re-sized.

Created the Cross-Fade looper.

Bug Fix: Wasnt correctly opening Wave Audio devices that support 16-Bit playback resolutions.

Added Audition Displayed and Audition Selected functions

Created the Demo Version.

Created Beta Tester Version.

Added Paste Merge

Added Cross-Fade Selected

Added Undo to virtually all destructive commands.

Added ToolTips to the .PAT file envelope shaper.

Added a Set Loop Point button to the LoopFinder.

Changed the file locations to store the .INI file in the Windows\System directory, and the .TMP files in wherever the user has their TEMP environment variable pointing to. This was necessary so that it will run from a CD ROM.

Bug Fix: Modified the Envelope Shaper to prevent the user from overlapping Envelope Points on the main Wave Viewer screen - this used to result in a rather ambiguous envelope manipulations.

Bug Fix: .PAT file envelope shaper wasnt scaling properly. It is now a lot easier to hit the offset ramp rails of 8 and 246, which should overcome problems with it creating quiet patches.

Created CD ROM Version.