****

Richmond Sound Design Ltd.
Theatre Sound Design, Show Control & Virtual Sound System Software

SoundMan-Server Release Notes
****

****

Version 1.0.148.0
2024-01-12

****

1) An incorrect code change was made in version 143 while clearing up warnings detected by the VS2022 compiler. This had the effect of making the command "SET GROUP G0" invalid, since the checking code now that that this was a recursive command. Changed the code back to what it should have been, less the warning.

2) Updated copyright dates to 2024.

****

Version 1.0.147.0
2023-06-19

****

1) The processor instruction features log messages were not properly formatted. They were missing an "I: " on the front of the message to show the information type.

2) After back-porting the bug fixes from the VS2022 source base, there were still a number of differences between the VS2010 and VS2022 source bases. Many of these were cosmetic, but some affected 64 bit code file operation. All of these have now been back-ported also, and the only differences now are in the warning messages being suppressed, since things in that area are different between VS2010 and VS2022.

****

Version 1.0.146.0
2023-06-14

****

1) Several code bugs were found in converting to Visual Studio 2022 due to its enhanced error checking. These fixes have been back-ported to the VS 2010 version of SMS in this change.

****

Version 1.0.145.0
2023-05-22

****

1) Updated the FFTW library to 3.3.10 from the earlier version that has been used.

2) In Selection synchronous event processing, a strange one-time hang happened when it thought another thread owned the lock, but the other thread thought that the synchronous event processing owned the lock. Added code to check for this odd circumstance, and continue normally if that is the case.

****

Version 1.0.144.0
2023-03-10

****

1) A recent change in how the TCLockMode enum was declared broke the code that tried to load old-format presets from much earlier versions of SMS.

2) This also broke the PRESET DESCRIBE code if the preset contained any track information for a loaded playback file.

3) This also broke RestoreTrackInfo when loading track information from an old preset.

4) Several releases back I added "V2" versions of some preset load and save routines to deal with new-format preset data structures. This release I have changed the names of the original routines to have "V1" on the end of the name. The new routines all have "V2" on the end of their names.

5) SMS used to default the number of playback channels to 16 (because that was the maximum on an old AudioBox) before SMS checked the license limits for the number of playbacks. Then SMS would load the interface using the license limits, which could be different than 16. This resulted in loading the interface twice, once with 16 playbacks and again with the correct number. We now only default the number of playbacks to 16 in AudioBox mode. This considerably speeds up the interface load in some cases.

6) When running in SCS mode we incorrectly showed the mode as "Show Cue System" rather than the correct "Show Cue Systems". This is now fixed.

7) Added PRESET SAVE ALL and PRESET RESTORE ALL to the Help command.

8) Removed MUTED from the list of options in the HELP PRESET SAVE command. MUTED can be specified for PRESET SAVE, but it will have no effect.

9) Added code to the locking routines used by selections to detect if a selection is locked by another thread when it is about to be deleted. This is a case that should never occur, but there are implications in dumps that it may occasionally occur. If this case is detected, the deleting thread waits until the class is unlocked. Log messages will be made of these events.

10) Previously SMS only saved the main dialog position when it exited cleanly. This made it difficult to reposition the window on screen and have the new position stick. Now we will save the window position any time it changes.

11) Added code to the OS version detector to detect Windows 11 and Server 2022.

12) The code added a release or two back to list the processor features available on the machine listed them one line at a time. On some processors this could be a very long list. The code has been changed to output multiple options on the same line, trimming lines at a little under 80 characters. This typically will only result in 3 or 4 lines of output, or less.

****

Version 1.0.143.0
2023-02-27

****

1) Changed assorted integer declarations to provide compatibility with a 64 bit compile of SMS.

2) Fixed the 16 bit conversion routines for ASIO input and output to work in both 32 and 64 bit modes. Previously they used MMX instructions that are only available in 32 bit mode.

3) Fixed a 32 bit pointer to 64 bits in resource access in initial startup.

4) Fixed an invalid argument crash that could happen in the OnTimer routine while debugging SMS.

5) The timer update routine formats all of the display items in the main window several times per second. Many of the small numbers were formatted using CString::Format. This resulted in thousands of small memory allocations on the heap. Some of these seem to get lost and result in memory leaks when we shut down. Reworked these to use sprintf_s instead of Format and eliminated all of the memory allocation and deallocation.

6) Reworked the ODS macro to take a variable number of arguments, simplifying debug code. Then cleaned up many calls to ODS to include the formatting.

7) If the interface shape changed and we had to rebuild the matrix, we failed to delete all of the matrix items from the lists of those items. Then when we tried to create new ones and add them to the lists, we failed because the list entry was not empty (it had an invalid pointer to a removed object). Changed the destructors for the various objects to make sure that they got removed from the various lists when they are destroyed.

8) Added a lot of debug and verification to AsioInterface::AddChannel and RemoveChannel to make sure that the arrays are being handled properly and make debug messages for tracking and error reporting.

9) Reworked the sample conversion path for 16 bit AIFF files to use SSE instructions. This should result in a decent reduction in conversion overhead.

10) Reworked the sample conversion path for 16 bit WAV files to use SSE instructions. This should result in a decent reduction in conversion overhead.

11) There were a lot of places that made a debug output of a log message just as they created it, in case the message took a long time to end up in the log. This was resulting in duplicate log messages in the debug output, since for a long time LOG_IT has called PostLogMessage, and PostLogMessage also does a debug output of any message it processes. Removed all of the extraneous debug output calls.

12) There were many direct calls to PostLogMessage. Changed these to LOG_IT calls for consistency. Once, long ago, there was a difference in what PostLogMessage and LOG_IT did, but they have been the same now for many years.

13) Changed LOG_IT (and PostLogMessage) to take a variable number of parameters and do printf-style formatting when required. Previously only a single string parameter was allowed, so if a log message needed to be formatted, it had to be done manually by the calling code. The new format greatly simplifies a lot of logging calls. Cleaned up these calls.

14) Increased the compile warning level from 3 to 4 (which I thought it had already been at), and had to add a bunch of casts and other stuff to get rid of the benign warnings that showed up.

15) The invalid parameter handler was inexplicably writing the error message to stdout, which nobody would see. Changed it to do a Message Box, then log a message about the invalid parameters, and finally take a program dump.

16) When the timecode reader value is stored in a preset, the freewheel count value was stored in a byte, but it really needed a 'short' to store the entire possible value range. This is now fixed with a version 2 of the timecode info storage for new presets. Fortunately the old format worked in most cases, so we can still restore the timecode info from existing presets.

17) Because the timecode freewheel count was stored in a UCHAR, an infinite freewheel, indicated by a value of -1, would have been stored, and thus restored, as 255. Which is not at all the desired result. Code has been added to the timecode info restoration routine to check for a freewheel count of 255 and treat it as -1. Of course there is also now a new storage format that correctly stores the timecode info in a new preset, but this change will make old ones restore correctly, assuming that the freewheel count was negative or less than 255. If it was 255 or greater (up to 500) then it will be restored incorrectly from an old preset, since data would have been lost when the value was stored.

18) Added code to check the processor capabilities for the new SSE4.1 instructions now being used. While these are new to SMS, they have been around on both Intel and AMD processors for ages, so using them on even very old machines should not be a problem.

19) Added SSE conversion for ASIO devices that use a packed 24 bit integer sample format. These are rare devices, but they do exist. The SSE conversion should be about 4 times faster than the sample at a time conversion that we did in the past. The SSE conversion will only be used on computers with the "SSSE3" instruction set option, which is most modern computers, but few older computers.

20) Defined "X64_BUILD" in Properties to make building 64 bit versions easier.

21) If a dump failed, the messages that printed the error code was misformatted.

****

Version 1.0.142.0
2023-01-31

****

1) Updated copyright messages to 2023. The previous version was supposed to do this, but somehow missed doing it.

2) Added a log message showing the sample rate we are attempting to set during the open of the selected ASIO driver. Previously we only showed the sample rate that we obtained on the open, and the driver can override the requested sample rate, so the two values may not be the same.

3) Fixed a bug in the 'configuration limits" message that incorrectly showed the requested sample rate.

4) Added the ability to limit the number of inputs and outputs on each interface given in an ASIOGROUP declaration. Also allow the selection of the specific channel numbers and channel number ranges to use.
The new syntax is:

```
CONFIG SET
      ASIOGROUP "name"
            {INTERFACE {"name" | number}
                  [{IN  | INPUTS}  {numchannels | channels | ranges | lists }
                  [{OUT | OUTPUTS} {numchannels | channels | ranges | lists }]}...
```

"numchannels" is the total number of channels you want to use from the interface. For instance "INPUTS 2" says you only want input channels 0 and 1.

A "channel" is a single channel number, for instance 0 or 3.

A "range" specifies a linear range of channel numbers. It is of the form <number> {-|TO} <number>. For instance "0-3" or "1 - 7" or "4 TO 5". The ending channel number in the range must be greater than the starting channel number. The range "0-3" indicates that you want channels 0, 1, 2, and 3, in that order.

A "list" is a collection of channel numbers and/or channel ranges, optionally separated by commas. For instance "0 1 2 3" or "0-3" or "5 2-3 4" or "5, 2 TO 3, 4". Any single channel number cannot appear more than once in the list. For instance "2 1-3" is an error, because the range 1 to 3 includes 2.

If INPUTS or OUTPUTS is followed by a single number, this is the total number of channels you wish to use from the interface. The channels will be used in the order they appear on the driver.

If INPUTS or OUTPUTS is followed by a list of numbers or channel ranges, only the listed channel numbers will be used, and they will be used in the order declared. A channel number may not appear more than once, either explicitly or as being a member of a range.

Either the number of inputs or outputs or both can be limited. The number of channels each interface will present is limited to the given value or the actual number of channels on the interface, whichever is less.

Inputs can be limited on one interface and not limited on another one in the same ASIOGROUP.

5) Added the ability to limit the number of inputs and outputs on each interface given in an INTERFACE declaration. Also allow the selection of the specific channel numbers and channel number ranges to use.
The new syntax is:

```
CONFIG SET
    INTERFACE       n|"name"
        INPUTS  {numchannels | channels | ranges | lists }
        OUTPUTS {numchannels | channels | ranges | lists }
        PLAYBACKS count
        [ANALOGFIRST | ADATFIRST | MADIFIRST]
        SAMPLERATE speed
        MODE {NORMAL | AUDIOBOX | ABEMULATION |
COMMANDCUE | SCS <key>}
```

"numchannels" is the total number of channels you want to use from the interface. For instance "INPUTS 2" says you only want input channels 0 and 1.

A "channel" is a single channel number, for instance 0 or 3.

A "range" specifies a linear range of channel numbers. It is of the form <number> {-|TO} <number>. For instance "0-3" or "1 - 7" or "4 TO 5". The ending channel number in the range must be greater than the starting channel number. The range "0-3" indicates that you want channels 0, 1, 2, and 3, in that order.

A "list" is a collection of channel numbers and/or channel ranges, optionally separated by commas. For instance "0 1 2 3" or "0-3" or "5 2-3 4" or "5, 2 TO 3, 4". Any single channel number cannot appear more than once in the list. For instance "2 1-3" is an error, because the range 1 to 3 includes 2.

If INPUTS or OUTPUTS is followed by a single number, this is the total number of channels you wish to use from the interface. The channels will be used in the order they appear on the driver.

If INPUTS or OUTPUTS is followed by a list of numbers or channel ranges, only the listed channel numbers will be used, and they will be used in the order declared. A channel number may not appear more than once, either explicitly or as being a member of a range.

Either the number of inputs or outputs or both can be limited. The number of channels each interface will present is limited to the given value or the actual number of channels on the interface, whichever is less.

6) An ASIO device is open when it is opened as the current interface, and also when it is part of an ASIOGROUP, even if that group is not currently open. A device can only be open in one use at a time. If you try to create an ASIOGROUP with a device already used in another group, or the currently open interface, the creation of the group will fail. This is not new behavior. What is new is the error message for the creation failure will now include the error code for the reason the creation failed.

7) Modified the "matrix shape" display that shows in the upper right of the main window when the interface is open to also show the current sample rate and buffer size.

8) Added an info log message when the initial preset load opens the ASIO driver.

9) Added a log message to SetSampleRate to show the sample rate being requested.

10) In SetParameters added several log messages about the sample rate.

11) Fixed formatting of a debug message in the code to create an ASIOGROUP. Without this change the debug code crashed SMS.

12) Added the driver error descriptions that can be returned when trying to open an ASIO driver to the values that can be returned by AsioClass::getErrorDescription. Previously these errors would have been described as "unknown".

13) Changed AsioClass::LoadAsioDriver to save the error value in a global value when attempting to load the driver. The global value can be accessed by the caller of this function.

14) Changed AsioMultiple::LoadAsioDriver to take the maximum number of inputs and outputs desired for the driver being loaded. This lets the number of used channels be limited.

15) Changed AsioDrivers::loadAsioDriver to return an ASIOError rather than a boolean error value. This lets it be specific about the reason for a failure.

16) Changed AsioDrivers::loadDriver to return an ASIOError rather than a boolean error value. This lets it be specific about the reason for a failure.

17) Changed AsioDrivers::getCurrentDriverName to return the driver error it gets when calling the lower-level function rather than a simple boolean.

18) Changed AsioDrivers::getDriverNames to handle the very improbable case that one of the ASIO drivers that are loaded will fail to return its name when we request it. In this case, the driver name will appear in the list as the error message we got from trying to get the name.

19) Added RequesteChannels class to hold the request type when desiring either specific channels or classes of channels. Also cleaned up the code for CONFIG SET INTERFACE to use the new class rather than the previous ad-hoc code.

20) If you attempt to CONFIG CLEAR ASIOGROUP <name> and the group is in use, you used to get an error message saying <name> was invalid. Now you will get an error message saying it is in use.

21) OnDisconnectGuts erroneously checked for command lockout and would fail if it was set. This routine is called in many places and cannot fail for command lockout. Check was removed. OnDisconnect, which calls OnDisconnectGuts, still checks for command lockout.

22) If you misspell INPUTS or OUTPUTS in an ASIOGROUP declaration, you would get a very unhelpful error message saying the ASIO interface was not open. This has been fixed to return an error showing the misspelled word.

23) Previously if you set up an ASIOGROUP where all interfaces had the same preferred buffer size, this size might not be used. Now if they are all the same, this size will be used.

24) ASIO interface channel ordering will now be saved and restored from a preset. This makes a new structure in the preset that will not be backwards compatible with older versions of SMS. However, if no specific channel ordering is given, the preset will be backward compatible.

25) Channel limits and ordering in an ASIOGROUP will now be saved and restored from a preset. This makes a new structure in the preset that will not be backwards compatible to older versions of SMS. However, if no specific channel ordering is given for any ASIOGROUP, the preset will be backward compatible.

26) When laying out VU meters on an interface that has a large number of channels, we can have more channels than we have room to display actual VU meters. In this case we lay out as many rows of meters as we have room for, then have a vertical scroll bar to scroll over all of the VU meters. Previously we assumed the SMS window was at the top of the screen, and calculated the number of rows based on the screen height. If the SMS window was not at the top of the screen, this over-calculated the number of rows of actual VU meters to create, and they extended off the bottom of the screen. Now we take the actual position of SMS on the screen into account and only lay out as many rows of VU meters as we can without going off the bottom of the screen.

27) If an invalid sample rate is specified on CONFIG SET INTERFACE, the error message displayed SAMPLERATE rather than the erroneous rate value. This is now fixed.

28) A new HELP command has been implemented. This can be used to list all of the commands that are valid in SMS, and will also give descriptions for many commands.

The format of this command is:

HELP [<tokens>...]

HELP by itself will list all of the top-level command words that are valid. You can then pick any of those terms, for instance CONFIG and enter HELP CONFIG. This will show all of the commands that are valid for CONFIG. This can be carried all the way down to the details of any command. A multi-line tree will be shown for all of the command levels. When the terminal item is listed in a command word sequence, a short description of that command might also be part of the response.

When executed from the TCP interface this command returns a multi-line response, where the final line is always a single dot (".") that indicates the end of the response. Responses can be moderately large, and control programs must plan for this if they want to issue HELP commands.

When executed from the command line, the response window will list all of the multi-line output on a single line. This is not useful, but if you click on the log window, you will see the correctly formatted response in the log.

29) The SET MATRIX command will now accept a GAINMIDI gain value as well as the previously available GAIN and GAINDB values. SET MATRIX ROW already accepted a GAINMIDI value.

30) The row input number given in SET MATRIX ROW previously had to be an integer for the command to work correctly. This only permitted setting row values for live input channels, not playbacks. There was a provision to allow the specification of channel names, but this code did not work correctly. Now an input row number can still be specified, or an input or playback row can be specified by giving the name of the input or playback channel. This can be a value like IN 1 or P0, or, if the channel has been named, the name of the channel can be given.

31) The SET CHANNEL <channel-spec> NAME command will now allow a name as a quoted string. However, the string can only contain uppercase and lowercase letters, numbers, and _ characters, and the first character must be alphabetic. The quoted name will be converted to all uppercase.

32) Fixed the splash screen that shows on initial startup of SMS to have the correct copyright date of 2023 rather than 2021.

33) Added a "/TRAYWAIT <seconds>" command line parameter. On some versions of Windows it can take several seconds after login before a tray icon can be successfully created. If this is happening when SMS is started from Task Scheduler at logon, try adding a /TRAYWAIT n parameter to the command line parameters. The value of "n" can be from 0 to 60 seconds.

34) Added a retry loop for tray icon creation. SMS will try for up to 30 seconds to create the tray icon before giving up. This may succeed in creating the tray icon without the need for a /TRAYWAIT command line parameter. Even with /TRAYWAIT, this up to 30 second retry loop will happen after the wait.

35) If the tray icon failed to create, SMS would queue messages to send to the tray icon, and they would remain queued forever, taking up increasing amounts of memory. Now there is a limit of 20 messages that can be queued for the tray icon when it has not been created. If more messages are queued the oldest messages will be deleted to keep the count at 20.

****

Version 1.0.141.0
2022-12-02

****

1) Updated copyright messages to 2023.

2) Changed the notify icon id number to be unique between S<M, SMA, and SMS. Perhaps this will prevent the occasional case where the shell taskbar icon fails to create.

3) Added code to the notification icon creation code to try to delete the current icon with the same ID before we try again to create the icon. Maybe this will give a better chance that recovery will work.

****

Version 1.0.140.0
2022-06-25

****

1) Improved the debug messages in AsioClass::LoadAsioDriver to show if the driver load failed, and give the driver index if it succeeded.

2) The logging for thread priority level now recognizes priority 20 as "Pro Audio".

3) There was code in AsioMultiple that assumed that only one AsioMultiple device would be created at a time. Added code to handle 'theDriverIndex' correctly when there are more than one AsioMultiple device.

****

Version 1.0.139.0
2022-06-20

****

1) Changed the non-debug ASSERT macro to pass the assertion test to the OurDebugAssert function so it can be displayed.

2) Changed OurDebugAssert to display the assertion text as well as the source file name and line number.

3) Changed OurDebugAssert to display a message box with the assertion text as well as calling OutputDebugString with it.

4) If SMS was started with a /WAIT parameter that had a delay time of less than 10 seconds or more than 30 seconds, SMS would crash on startup. This has been fixed.

5) Fixed a problem where "SET CHAN I0-1 GEQ OCTAVE 1 GAINDB -5.2" was parsed incorrectly and a syntax error generated.

6) Changed the number of static instances of AsioClass (used to hold ASIO driver info) from 16 to 32. A customer already had 16 real ASIO drivers on his system, which meant the first AsioMultiple instance created by SCS was one more than the table of drivers could hold.

7) Added "theDriverIndex" and the AsioClass class pointer to several debug calls to see where an invalid index might be coming from.

8) Fixed a debug message in AsioMultiple::ASIOCreateBuffersEx that didn't have the necessary parameter info, so it printed garbage.

****

Version 1.0.138.0
2022-06-10

****

This is a debug-only release, sent to the one SCS customer that is having a problem with not being able to open a particular device.

1) Changed copyright dates to 2022.

2) Added a great number of debug messages to AsioMultiple::CreateBuffersEx and related functions to try to track down an ASE_NotPresent (-1000) return value when trying to create a driver in SCS. The existing debug gave no clue, which seemed to indicate an early bailout for an unexpected condition like a driver array bounds error. The driver we are trying to create was number 12.

3) Added debug to AsioClass::ASIOCreateBuffersEx to catch the error returns.

4) Added debug to AsioMultiple::LoadAsioDriver to catch the error returns.

5) Added debug to AsioMultiple::RemoveAsioDriver to catch the error return.

6) Changed the word "called" to "entered" in the debug at the top of the AsioMultiple functions. I was getting confused with "called" being at the call site and not the start of the procedure.

****

Version 1.0.137.0
2021 Never released

****

1) Allow GET CHAN GEQ OCTAVE RESPONSE to get the total response for all octave bands. This is similar to GET CHAN GEQ RESPONSE to get the response in bands.

2) Allow GET CHAN GEQ BAND RESPONSE as a synonym for GET CHAN GEQ RESPONSE to match the syntax for getting the overall octave response. The old syntax remains functional.

3) Previously it was possible to set multiple bands or octaves to a single gain value by using SET CHAN GEQ BANDS|OCTAVES number -|TO number. However, you needed to use the plural BANDS or OCTAVES. If you entered SET CHAN GEQ BAND|OCTAVE number -|TO number you got a syntax error on the dash or TO, as only a single band or octave number was expected. Now multiple bands or octaves can be specified using BAND or OCTAVE without the plural. The plural syntax remains functional.

4) Computing the frequency gain parameters when setting gain in octaves did not work correctly in all cases. This has been corrected.

****

Version 1.0.136.0
2021-11-04

****

1) Fixed a small formatting bug in the debug output for displaying the contents of a preset.

2) When attempting to restore the startup preset, if the startup preset does not contain ASIO information, the load of the preset will fail. There are now specific log messages indicating that the load failed for missing ASIO information in the preset.

3) When creating a new startup preset with the PRESET SAVE command, it is mandatory to save the ASIO connection info. Previously it was incumbent on the user to remember to add a CONNECTION token to the list of items to be saved for the startup preset. Now code has been added to check if the name of the preset being saved is SOUNDMAN_STARTUP, and if it is, make sure that there is a CONNECTION token in the save string. If one is added an informational log message will be generated as the preset is saved.

4) Added log messages when the main window is minimized or restored.

5) Added and cleaned up some documentation in the ThirdOctaveEQ code.

6) Fixed the parsing of the SCRIPTFILE command to work correctly if the "AS scriptname" part of the command was not supplied.

7) Reformatted and enhanced the documentation in CommandParser.h. Made the token access commands public.

8) A malformed SCRIPT file that was missing the script end line could end up leaking memory when an attempt was made to load it. This has been fixed.

9) A script can now interrogate the graphics EQ band frequencies, gains, and band gains in dB for all channels. Previously only values for parametric EQ could be obtained by a script.

   The general format for doing this is:

   ASSIGN|LET variable = CHAN channel-identifier <item>

   These can also be used in IF statements rather than assigning them to a variable:

   IF CHAN channel-identifier <item> GOTO label-name
   IF CHAN channel-identifier <item> THEN assignment-statement

   Where <item> is one of the following:

   GEQ BAND n // band number 1 to 31
   FREQ // band center frequency
   GAIN // band gain, 1.0 = flat
   GAINDB // band gain in dB, 0.0 = flat

10) A script can now interrogate the overall EQ gain for a channel at a particular frequency, or the gain for just the graphics EQ at a given frequency, or the gain for a parametric band at a given frequency, or at the band center frequency, and can interrogate the gain for a graphics EQ band at the band center frequency.

The format is the same as above for assignment statements and IF statements.
The format of <item> is:

```
EQ RESPONSE AT freq             // overall eq response at frequency
EQ BAND n RESPONSE              // band response at center/corner freq
EQ BAND n RESPONSE AT freq      // band response at given frequency
GEQ RESPONSE AT freq            // graphics response at frequency
GEQ BAND n RESPONSE                     // graphics response at band center
```

11) The routine to get the current EQ response in dB erroneously returned a response of 1 dB rather than 0 if a bad value was requested.

12) Several of the script routines to get EQ values from channels were returning incorrect response values.

13) Several "safe" Microsoft sprintf_s statements could end up corrupting memory, when the "unsafe" original sprintf statements would not have. Changed code around to eliminate the unsafe code.

14) When an initial preset successfully set the ASIO interface, the focus was left on the ASIO selection droplist in the main window rather than being moved to the command line as it should have. The focus is now properly set to the command line.

15) There are now commands to set and get the frequency response of the graphics third octave equalizer on each channel by octaves as well as third octaves.
The difference in the command structure is that a BAND is a third of an octave, and are numbered from 1 to 31. An OCTAVE is an octave, and are numbered from 1 to 10. OCTAVE 1 corresponds in frequency to BAND 3, and each subsequent octave is 3 bands higher than the previous one.

The octave center frequencies are:

Octave	Frequency
1	31
2	62
3	125
4	250
5	500
6	1000
7	2000
8	4000
9	8000
10	16000

Other than the convenience of using octave numbers rather than scaled band numbers, there is no difference in the GET commands. However, when setting

the gain of an octave, the gain is smoothed across three bands to the next higher and lower octaves, whereas when setting by bands the gain is only smoothed to the next higher and lower bands.

The syntax of the graphics EQ GET command is now:

```
GET CHAN|CHANNEL channel-spec
        GEQ | GRAPHICEQ
                QUAL | QUALITY          # GrEqQuality     chanid=number
                DELAY                   # GrEqDelay       chanid=fpt
                ENABLE                          # GrEq              chanid=ON|OFF
                GAIN                    # GrEqGain        chanid=fpt,fpt... 31 times
                GAINDB | RESPONSE       # GrEqGaindB      chanid=fpt,fpt,.. 31 times
                GAINDB | RESPONSE AT freq
# GrEqFreqGaindB chanid=fpt,fpt freq,gain
                OCTAVE number           #
                        GAIN            # GrEqOctaveGain          chanid=band=fpt
                        GAINDB | RESPONSE
# GrEqOctaveGaindB      chanid=band=fpt
                BAND number             #
                        GAIN            # GrEqBandGain            chanid=band=fpt
                        GAINDB | RESPONSE
# GrEqBandGaindB        chanid=band=fpt
```

And the syntax of the graphics EQ SET command is:

```
SET CHAN|CHANNEL channel-spec
        GEQ | GRAPHICEQ
                QUAL | QUALITY 1..8
                ON|OFF
                OCTAVE int              # by octave number 1..10
                        GAIN    fpt
                        GAINDB          fpt
                        =       fpt     # gaindb
                OCTAVES int [{TO|-} int] {= | GAINDB} fpt...
# multiple bands at once
                BAND int                # by band number 1..31
                        GAIN    fpt
                        GAINDB          fpt
                        =       fpt     # gaindb
                BANDS int [{TO|-} int] {= | GAINDB} fpt...
# multiple bands at once
                FREQ | FREQUENCY        # by frequency 20..20000
                        GAIN    fpt
                        GAINDB          fpt
                        =       fpt     # gaindb
```

16) Commands originating from scripts now have a comment added to the end of the command in the log of "Script <scriptname>". This makes the source of the command easier to locate when there are multiple command sources.

17) The machine id and dongle id windows in the main dialog are no longer incorrectly flagged as tab stops.

18) Attempting to start SMS from the windows Scheduler immediately after boot may fail to load an initial preset if the ASIO driver and card have not

been able to initialize by that point in time. To avoid this startup failure, there is now a command-line parameter to SMS to cause it to delay before attempting to find the dongle or the ASIO drivers, and load the initial preset.

/WAIT <seconds> // seconds = 0 to 60

This command-line parameter passed to SMS at startup will cause an initial delay of the specified number of seconds. Note that the maximum delay is one minute. Historically values on the order of 10 to 20 seconds have been sufficient for most ASIO drivers to get their house in order and be ready to run.

A log message will be created at the start of this delay, and the tray icon will also make a popup balloon showing that SMS is waiting for this initial delay. The "Initializing" screen that blanks the main interface during the dongle search will also show "nn Second Delay" instead of "Initializing" during the initial delay. The "nn" will be an incrementing number of seconds during the delay. This won't be exceptionally useful on an unattended system, but it is handy to make sure that a specified delay is being acted upon when setting up the initial command line.

19) If the WAIT time in a script was too long or too short the not terribly helpful "Invalid wait time" error message used to be produced. There are now separate error messages for a too long and too short time that explicitly state the minimum and maximum valid times.

20) If the WAIT time in a script was less or equal to 10ms it was considered invalid. This is changed to be only less than 10ms.

21) Computing the fade time for a playback speed change was slightly incorrect and could result in the speed change overshooting the target value. In the extreme case of a large speed reduction and a file with a very low sample rate, the speed could go negative, resulting in the track stopping. This has been corrected.

22) The command line in the main window is now a droplist rather than a plain edit window. The last 40 commands are saved as a command history. The user can use the up and down arrows to go to the previous or next command in the history, or can use the mouse to select a command from the history. Previously a command could be selected from the output window to be re-executed, and this is still possible. However, commands can easily become lost amongst the output lines, so the command window makes it easier to reuse a sequence of commands.

23) The output for CONFIG GET SCRIPTS had a minor formatting error where the status was not separated from the script name by at least one space. Now the longest script name is detected, and all of the comments align in a column when the script names and status are listed. The colon following the script name in the output has also been eliminated.

24) In a script, testing a playback channel to see if it is stopped after playing did not work correctly. This was fixed by separating the concepts of having media loaded on the channel and being at the end of the playback range.

25) Testing playing and stopped status of a playback channel in a script messed up because the tests were reversed. This is fixed.

26) When the PRESET DESCRIBE code was added in version 134, a small error was made that made it impossible to load older presets, because the size of a data structure in the preset accidentally changed. Code has been added to accept either the old or new size of this structure, again allowing older presets to load correctly.

27) If a preset contained data for many groups and we only asked to restore a subset of those groups, the code to do the restore would get locked in a loop. This has been corrected.

28) The debug code in PRESET DESCRIBE to describe the group contents of a preset was incorrect, and skipped over the group info.

****

Version 1.0.135.0
2021-06-09

****

1) Repartitioned the OnDisconnect code so that we can do some button-specific processing before handling the main interface disconnect logic.

2) We now make a specific information log record when the DISCONNECT button is pressed on the user interface.

3) If the interface is not open when the DISCONNECT button is pressed, we make a log record indicating that the press had no effect and do nothing more.

4) If the interface is open and the operating mode is SCS when the DISCONNECT interface button is clicked, we do a popup asking the user if he really wants to manually disconnect the interface in SCS mode, pointing out that SCS should be the only one doing this, and doing it manually will confuse SCS. If the user chooses to continue we log a warning that the user manually closed the interface out from under SCS. If the user chooses wisely to not close the interface, we log that too.

5) If we are operating in SCS mode and an attempt is made to select an interface with the droplist and CONNECT button, a popup message informs the user that the interface can only be selected by SCS itself. A log message is made that a manual attempt to set the interface was ignored, and no further action is taken on the connection attempt.

6) When a connection is made manually using the droplist and CONNECT button a CONFIG SET INTERFACE command is generated. This command now has a comment of "Dropdown driver selection" to indicate that the interface selection was made from the droplist.

7) If the CONNECT button is clicked with no driver selected, the click used to be ignored. Now there is a popup message instructing the user to select a driver from the droplist before clicking the CONNECT button.

8) In command lockout mode, the CONNECT button will be completely ignored. Previously if SMA was running it could bring up a popup message and then the configuration dialog in SMA, which was incorrect in command lockout mode. The connection attempt and rejection will now also be logged.

9) In command lockout mode a DISCONNECT attempt was correctly ignored, but the log message showing an attempt in command lockout mode did not indicate what control panel action was ignored. Now it indicates that it was a disconnect attempt.

10) When attempting to change the driver selection in the droplist in command lockout mode, the change was ignored and logged, but the log message did not indicate which interface command was ignored. Now it indicates that it was a driver change.

11) If an attempt is made to change the driver interface in SCS mode, the change is ignored, and a log message is made indicating that a manual

attempt was made to change the driver selection in SCS mode.

12) Attempting to exit SMS when command lockout mode was enabled was ignored and a log message generated, but it did not indicate the command that was ignored. Now the log message will indicate than an app exit attempt was ignored.

13) When a command is entered manually in the command window, and it does not already have any comment text on the command, a comment of "From cmd window" will be added to the end of the command before it is processed. This will make it easier to distinguish manual commands entered by the user from commands entered through the Telnet interface from some controlling program such as SMS or SMA.

14) The 'interface changed' flag is set when a SET command is seen. If this flag is set at the end of command processing, we update the VU meter layout in the main window. It was cleared in SET INTERFACE before any changes were made, and then set again if something had changed. This way the meters were not updated if the same SET INTERFACE command was entered twice, since there would be no changes the second time it was entered.

However, if an error was seen while processing a second SET INTERFACE command when the interface was already open, the 'interface changed' flag was still set, even though no change had been made because the command errored out. The flag should have been cleared before the command parsing checked for possible errors, and not after the parsing but before any changes were made. This has now been adjusted, so extra work is not done if a second invalid SET INTERFACE command is entered while the interface is already open.

****

Version 1.0.134.0
2021-05-20

****

1) Cleaned up some errors in the comments describing how the preset functions work.

2) Made sure to initialize all fields in the PresetInfo structure when it is created. Previously a couple were not initialized. This didn't cause a problem, but it was messy and could eventually have been a problem.

3) Added a debug routine to the Preset code to decode the preset token type into a name string rather than a number. This is used in debug logic that decodes presets.

4) Moved many of the Preset data structures from various locations where the data is created to the PresetInfo.h file so that all of the preset structures are defined in one place.

5) Added a routine to describe an entire preset into a CSV text file that can be loaded into Excel or some other spreadsheet. Currently this command will create a file with info on all channels that have a gain greater than -144dB, or have parametric EQ values or a nonzero delay. Other non-null channel information may be added in future versions.

    PRESET DESCRIBE "preset file name"

    This command will create a csv file with the same name as the preset in the same location as the preset file. If no path is given on the preset file name, the preset file will be expected to be in the presets folder. In the unlikely event that the preset name itself has the suffix "csv", the decoded preset file will have the suffix "csv1".

    The csv file has the channel name in the first column, the type of data in the second column, often the mute or enable status for the item in the third column, and the information about the item in subsequent columns.

6) Created a new PresetDescribe.cpp source file to contain the code for the PRESET DESCRIBE command.

****

Version 1.0.133.0
2021-01-22

****

1) Removed the assert failure that caused a dump and crash if the ASIO driver returned a zero for the preferred buffer size during driver open. Instead the invalid size is checked and an appropriate log message created, then the open is failed for an invalid parameter error.

****

Version 1.0.132.0
2020-12-04

****

1) Fixed some comment formatting in ThreadManager.cpp.

2) Added some runtime debug and log output to ThreadPriority.cpp to make sure we create useful debug information if thread priority calls fail. Previously we only got this debug information in a debug build (not a release build), and the debug information lacked some information needed in a release build.

3) We did not correrctly decode the Windows OS version on Windows 10. In OsVersion.cpp, added a function to call RtlGetVersion to get the real OS version on Windows 10 systems.

4) In OsVersion.cpp fixed some bad indenting of the main function and add some descriptive information.

5) In OsVersion.cpp, adjusted the code to determine if we successfully determined the OS version, and if not, print the numbers in the log message that are used to determine the Windows version so that the value can be determined by hand from the log information.

6) In OsVersion.cpp, added code to handle more recent OS version encodings to properly decode Windows 10, 8.1, and 8.

7) Adjusted the debug log messages that give the times from ASIOInit to various driver calls to line up all of the timestamps in the messages. Also indented the message text by one character to make reading the log easier.

8) When using only one worker thread, say "1 thread" rather than "1 threads" in the log message.

9) Sorted the thread information array in ascending order to make the debug messages on thread priority handling easier to track.

10) BumpAsioDriverPriority was passed a DWORD array of thread prioritry values, but priorities are integer values that may be negative. This resulted in incorrect comparisons when the priority was negative. Defined a CIntArray type and change to pass integer array values.

11) The OsVersion.cpp code incorrectly thought that the OS info it was requesting was in ASCII, but the data returned was wide character. This could have resulted in buffer overrun and data corruption. Fixed things to understand that wide characters were being used. This also resulted in the service pack information not being correctly formatted in the Windows version string that was created.

12) If the AV thread scheduling call fails to raise the priority of the driver callback thread (as is often the case), we now make a second attempt using SetThreadPriority before giving up on being able to adjust the situation.

13) If the AV thread scheduling call fails to lower the priority of the driver callback thread when we are closing the driver (as is often the case), we now make a second attempt using SetThreadPriority before giving up on being able to adjust the situation.

14) The previous version of SMS added some debug code that output a message when a dump was taken. This message served it's purpose and has been removed since it just cluttered up the log.

15) The code added in the last version of SMS to limit ASIO timeout dumps to no more than one a minute had a bug and didn't work correctly. This is fixed.

16) Added code to not take more than five ASIO timeout dumps in an hour. If the ASIO device is acting up on an unattended long-running system we don't want to fill up disk with probably-useless dump files.

****

Version 1.0.131.0
2020-12-03

****

1) Seemingly the dialog pointer was wrong when logging a dropout in the ASIO callback timing. This probably indicates memory corruption, but it was causing a secondary unrecoverable crash. Changed the code to pass the address of the dialog directly to the logging code to try to get around this. This was only happening when a failing driver was having many repeated dropouts in the sequence of ASIO callbacks.

2) Added code when taking a dump because of an ASIO timeout to reset the current timestamp to after the end of taking the dump. This will prevent the possibility of multiple dumps because the dump process takes a long time.

3) Added code to take at most one dump a minute due to the ASIO callback running too slowly.

4) Changed PostLogMessage to do debug output of the log message before putting the message in the message queue, so we will see the debug message in a timely manner even if the queue gets backed up.

5) Changed asioMon to not do debug output of the messages it generates after logging them, since PostLogMessage already does this debug output and this was resulting in duplicate messages being output to the debug output.

6) Changed the declaration of several AsioClass procedures that returned strings to return constant strings. This prevents an accidental assignment to the procedure return value in an if statement, when an equal comparison was intended.

7) The Focusrite USB ASIO makes no attempt to deliver ASIO sample events on an even clock. It appears to get a buffer of data from the USB interface and then spill those samples into SMS as fast as it can, then go back and wait 10 or more ms for the next sample buffer. The result is that SMS would see and report an overtime buffer every few buffers. There is now a check for the Focusrite USB ASIO interface, and code to average one ASIO buffer's worth of sample periods when an overtime period is seen. Only if the average of these periods is still overtime will an overtime period be reported. Other drivers will still report overtime ASIO periods as they happen.

****

Version 1.0.130.0
2020-11-25

****

1) Script parsing code testing for a date type of LEAPYEAR was incorrect.

2) Added a "channel testable item" of VU to channels in scripts. This lets a script check the current level of a channel.

3) Found and fixed a bug in variable evaluation in a script. Values of type 'double' were being treated as integers and losing fractional significance.

4) The script handler would log a script waking up from a wait no matter how short the wait was. For a script in a monitor loop with a short wait this polluted the log with a lot of useless entries. Now a wakeup will only be logged for a wait of 10 seconds or more.

5) The script WAITFOR command waiting for a channel testable item (such as waiting for the gain to change to a certain value) had the test sense reversed, and wasn't waiting when it should have.

6) A WAITFOR command would incorrectly chew up all the processor time while waiting for an event to occur.

This was created for a user who wanted a way to detect a level on an input channel in SoundMan. They needed to duck other inputs if there is any audio on it.
To do this requires writing and running an SMS script that sends a command to SMS to start (and possibly stop, if you want) the script. The script can't be started until the ASIO interface is open.
The script will sit in a loop checking for output level to show up on your favorite channel. When it sees the audio show up, it can proceed to duck the other channels. Then it needs to sit around for a while to see when audio goes away, and un-duck the other channels. Then it can loop back and wait for more input.
Something like the following will work:

```
script DuckAudio begin

        variable mic_vu ; current mic output
set chan i1000 x1000.0-1 o1 mute off gain 1

label waitForAudio

        ;       Wait until we see some VU from the mic

        log user "waiting for audio on chan o0"
waitfor chan o0 vu > 0.1

        ;       Log the audio level on the mic channel

        assign mic_vu = chan o0 vu
log user "saw @mic_vu VU on chan o0"

        ;       Duck the audio on the other mic channels

        set chan o1 gaindb -25 fadetime .5
```

```
        ;       Now wait for the guy to stop talking.

        log user "waiting for NO audio on chan o0"
waitfor chan o0 vu < 0.1
log user "no longer talking, unducking o1"

        ;       Unduck the audio on the other mic channels

        set chan o1 gaindb 0 fadetime .5 ; unduck the other mic channels
goto waitForAudio ; channel has gone quiet, wait for more talking

script DuckAudio end
```

Save this as a text file in the SoundMan Scripts directory, or anyplace you want to put it.
To run this you can do one of two things:

```
        load scriptfile "DuckAudio.txt"     ;       this is the name of the file
        run DuckAudio                       ;       this is the name on the "script" line in the file
```

or

```
        run scriptfile "DuckAudio.txt"      ;       this both loads and runs the script file
```

If you want to stop the script execution you can

```
        cancel DuckAudio
```

The script above has lots of handy debugging stuff in it; all those LOG lines.
You don't need (or probably want) that stuff in a production script, so things will simplify considerably:

```
script DuckAudio begin

        ;       Initial channel setup so I don't have to do it myself.

        set chan i1000 x1000.0-1 o1 mute off gain 1

label waitForAudio

        ;       Wait until we see some VU from the mic

        waitfor chan o0 vu > 0.1

        ;       Duck the audio on the other mic channels

        set chan o1 gaindb -25 fadetime .5

        ;       Now wait for the guy to stop talking.

        waitfor chan o0 vu < 0.1

        ;       Unduck the audio on the other mic channels

        set chan o1 gaindb 0 fadetime .5    ;       unduck the other mic channels
goto waitForAudio                   ;       channel has gone quiet, wait for more talking

script DuckAudio end
```

The above uses VU on output channels, but it will work on inputs and crosspoints too.

****

Version 1.0.129.0
2020-11-19

****

1) Updated splash screen copyright to 2021.

2) Updated dialog and text string copyrights to 2021.

****

Version 1.0.128.0
2020-03-12

****

1) Fixed a problem that occurred when attempting to open the Focusrite USB ASIO device when the USB cable was not connected. A bug in the Focusrite driver caused an exception.

2) The Focusrite USB ASIO driver actually has a fixed minimum granularity of 16, but the driver erroneously claims 8. A workaround has been placed in the driver open path to fudge the value to be correct. This was previously fixed for the multiple ASIO driver path used by SCS, and now fixed for the normal driver open path.

****

Version 1.0.127.0
2020-02-05

****

1) Corrected a warning log message for the Focusrite driver that was badly formatted.

2) The previous hack code to set the Focusrite driver to binary was incorrect. New information from Focusrite indicates that it is a fixed granularity of 16, and the error was reporting the minimum size as 8 rather than 16.

3) Fixed a bug in AsioMultiple code that was putting the wrong driver name in various log and debug messages.

4) Removed extraneous newlines from the end of several driver-related log messages.

5) Fixed an ASIO driver cleanup ordering problem that caused a null pointer reference during shutdown.

6) Fixed a shutdown problem where we got heap corruption by trying to create one final log message after the main dialog had been destroyed.

7) Fixed several problems that would leave the script runner thread running at program termination, leading to a small temporary memory leak.

8) Previously CONFIG SET SHUTDOWN NOW actually put the PC to sleep rather than shutting it down. Also, there was debug code that would make a beep in the audio output. CONFIG SET SHUTDOWN NOW will now attempt to immediately shut down the PC running SMS. It will not wait for other apps to close, and will not automatically restart after the shutdown.

9) A new CONFIG SET SLEEP NOW command has been added to attempt to put the system to sleep. This is the same function previously performed by the incorrectly named CONFIG SET SHUTDOWN NOW command. The debug beep has been removed from this function.

10) Added an invalid parameter handler function to take a dump in the case of an invalid parameter to a system call. Without this override, Windows 10 will call ExitProcess, silently making the program vanish with no possibility of getting any debug information on what went wrong.

11) Determining the EQ shape from the coefficients given by ABShowMaker did not work well due to rounding errors in the ABSM computations. This caused peaking filters to be incorrectly recognized as low shelf filters. A small epsilon value was added to the value comparisons to allow numbers that only differed by a small amount to be treated as equal.

12) Now display the EQ gain info in dB as well as a gain multiplier in the EQ debug output message.

****

Version 1.0.126.0
2020-01-31

****

1) Adjusted internal documentation on SET SHUTDOWN NOW and acquiring the system shutdown privilege.

2) Added code to AsioMultiple to check for an invalid granularity specification from a driver and either set the driver info to fixed granularity, or if it is the Focusrite USB ASIO driver, set it to binary.

****

Version 1.0.125.0
2020-01-29

****

1) When importing a show from ABShowMaker and then trying to run it, the order of playback setup events is different than that generated by ABEdit. This resulted in looping tracks not correctly looping. Changes were made to track initialization to correct this.

****

Version 1.0.124.0
2020-01-26

****

1) Added debug output code to show the call parameters to the ASIO driver ASIOCreateBuffers call before making the call, in case the call fails.

2) Wrapped the ASIOCreateBuffers call in a try/catch statement to try to protect SMS against a fatal crash inside the driver code. Unfortunately this still does not help for a Focusrite driver.

3) Added code to try to take a dump in case the ASIOCreateBuffers call fails.

4) Updated copyright dates to include 2020.

****

Version 1.0.123.0
2019-12-02

****

1) Corrected a time display in messages displaying slow ASIO driver startup.

2) Corrected the display wait time on driver startup to not get a whole lot of messages quickly about a slow driver startup.

****

Version 1.0.122.0
2019-11-24

****

1) Changed the ASIO callback monitoring code to handle waiting for an initial callback differently. There are indications that some new USB-based ASIO drivers can take an extensive period (perhaps more than a second) before they make the first callback. This was resulting in a dump on every driver startup. We now wait up to two seconds for the initial callback without taking a dump, but making a message every 100ms if the driver has not started by then. After two seconds we will go into normal monitoring and take a dump.

****

Version 1.0.121.0
2019-09-08

****

1) A text buffer in the dump routine was too small, possibly resulting in a secondary fault while trying to take a crashdump.

2) The program dump code has been failing when trying to take a program dump when requested. The parameters to the dump request were changed to match those used by SoundMan Monitor, which has been working.

3) Commented out an unused routine that was calling GetThreadId, which is not implemented on Windows XP Embedded. How this has been working for at least four years without complaints before this I don't know.

4) Commented out an extraneous debug message that was generated on every log message.

****

Version 1.0.120.0
2019-06-22

****

1) A cleanup ordering problem could result in destroying the log lock before the last log messages were created. The type of lock was changed to not use crt allocated memory and avoid this problem.

****

Version 1.0.119.0
2019-06-22

****

1) If we are in the realtime priority class at startup, we avoid changing the process priority at any point, since on Win7 and beyond we will lose the realtime attribute and not be able to get it back.

2) Added debug code to track changes in process priority class and changes in thread priorities.

3) Disabled an assert in InputConverter::ClearSelection that was tripping when it shouldn't have been.

****

Version 1.0.118.0
2019-05-20

****

1) Added an attempt to flush pending log messages when we get an external dump request.

2) Added code to the dump code to try to flush pending log messages before taking the dump.

3) Added code the lock object to make a log record and take a dump if we decrement the recursion count negative, since this should be impossible.

4) The lock destructor didn't always delete the associated critical section object. This has been fixed.

5) Created an interlocked list search routine for the DispatchList class. This should eliminate the rare problem with getting a bad pointer to a Selection.

6) Added a warning disable to stdafx.h for a useless warning in a Microsoft header file at xmmintrin.h line 386.

7) Changed things so the ASSERT macro will work in release build code.

8) Fixed a missing initialization in the TRYLOCK constructor that could occasionally cause locking to fail.

9) Rewrote the synchronous event processing loop to remove a possible deadlock condition.

****

Version 1.0.117.0
2019-03-12

****

1) Updated copyright to 2019.

2) Added dump catch logic around setting up a multiple-ASIO device. There seem to be problems with quietly crashing during the setup.

****

Version 1.0.116.0
2018-07-16

****

1) Fix a possibility that we could lock the ASIO class lock multiple times while processing a synchronous event.

****

Version 1.0.115.0
2018-04-20

****

1) Fixed a crash that could potentially occur while moving a selection from the ready list back to the idle list.

2) Make sure we don't get hung up in the ASIO thread waiting for a synchronous event to complete.

3) Make sure we don't think we own a lock when it is not owned by anyone.

****

Version 1.0.114.0
2018-02-13

****

1) Fixed a deadlock that could occur on a multiprocessor system where SMS could hang as a result of some command sequences sent by SCS if the selections being played were exactly the right length.

2) Updated the copyright notice to include 2018.

****

Version 1.0.113.0
2017-10-02

****

1) Added code to TcpSMSCommandHandler::OnReceive to try to keep it from hanging if the command port is closed immediately after sending a command to SMS. This may or may not work, this is a test version.

****

Version 1.0.112.0
2017-08-03

****

1) Changed debug log messages about a hung ASIO callback thread into warning messages so that they will extract with other warning messages when using the DOS "FIND" command on log files.

2) If you set up a loop and then canceled the loop before it repeated once and also set a new stop time, SMS would have jumped from the original loop end time directly to the stop time and stopped. If the loop had played at least once it would have worked correctly, playing through to the new stop point and stopping. This is now corrected so that it will place through to the new stop point whether or not the loop has played at least once.

3) Logged messages were showing up twice in the debug output, which was confusing. This is corrected.

4) If for some reason (like a disk retry) a buffer was still marked as reading when we needed to output the data from the buffer, we would make one warning message for every sample in the buffer until the read completed or we used up all of the samples in the buffer. This slowed things down and could lead to a dump because the system appeared unresponsive. Now we only give one message for each buffer when we first notice it reading and a second one if we notice that the read has completed while we still are using the buffer.

****

Version 1.0.111.0
2017-02-17

****

1) The Ecoustic version seemed to be having problems with the different method of callback handling. Reverted the callback handling to the normal method.

2) Updated copyright to 2017.

****

Version 1.0.110.0
2016-12-26

****

1) Long multiline command responses, such as the response to CONFIG GET CHANNELINFO, CONFIG GET SCRIPTS, or CONFIG GET SCRIPTFILES, could crash SMS. This has been fixed.

****

Version 1.0.109.0
2016-05-26

****

1) Removed the SafeSetWindowText function now that the root cause of the problem is fixed and this workaround is no longer needed.

2) It was possible to crash SMS by changing graphic EQ parameters when you specified ON or QUALITY when setting individual band parameters. This should now be fixed.

****

Version 1.0.108.0
2016-05-01

****

1) Problems creating the VU meters for large channel configurations have been fixed. The VU meter display has been re-enabled.

2) On rare conditions when closing the ASIOSMS driver we could crash while getting a log message time stamp. This should now be fixed.

3) Creating a large number of channels created a large number of Windows events that were rarely used. Changed the code so that events are only created when they are actually needed.

4) Added a lot of debug timing code to track down a problem with a specific ASIO driver. This code is normally disabled, but can be used when needed.

5) Changed the ASIOgroup interface to not call ASIOGetSamplePosition unless absolutely necessary. Some drivers do not respond to this callback in a timely manner and can result in audio glitches.

6) Changed the ASIOgroup driver to only pass an ASIOGetSamplePosition call to the first driver in the list rather than requesting the sample position information from all drivers unless one returns an error result. This will produce a more consistent timing value and avoid a problem in the RADAR driver if it is not the first driver.

7) Changed the ASIOgroup driver to always pass along the timecode information on BufferSwitch from the first driver in the group, rather than the last driver to check in on each buffer switch as was previously done. This will present a consistent timecode picture in case the user wants to know that.

8) The preset saving code was using the incorrect variable for the ASIO group count, and so failing to save the ASIO groups. This would prevent restoring a connection to an ASIO group on startup.

9) The conversion from VC6 to VS2010 involved changing a number of strcpy commands to the new "safer" MS-defined "strcpy_s" commands. In a number of cases these commands have corrupted memory, resulting in various crashes and unpredictable behavior. Hopefully all of them have now been located and converted back to actually safe string copies.

10) Changed AudioChannel::SetName to not set the name if the new name was the same as the old name. This drastically sped up opening the interface from a preset when there were a large number of channels.

11) When loading an ASIO group driver by name, the name comparison is no longer case sensitive. A side effect of this is that you can not have multiple groups that differ only in the letter case in the names.

12) When clearing an existing ASIO group, the name to clear is no longer case-sensitive.

13) If SMS is attached to a debugger it will not take a dump if an overtime ASIO callback is detected. Overtime callbacks will occur any time the debugger stops the program for any reason.

****

Version 1.0.107.0
2016-04-11

****

1) Removed extraneous newlines from some new log messages.

2) Added some more debug to try to track down why CreateBuffers is failing on one particular driver.

3) Fixed a problem where the timecode reader would stop decoding timecode values after five days of continuously running timecode.

4) Added code to debug messages in Multiple ASIO callbacks to display the driver name as well in the driver index number.

5) Removed extraneous newlines from end of timecode reader periodic status log messages.

6) Added code to the thread status debug messages to show the thread affinity.

7) Added a startup debug routine to the multiple ASIO device driver to log the arrival times of the BufferSwitch callbacks from all of the drivers for the first two seconds or so after startup.

8) Temporarily disabled the VU meter display, as it was causing problems with a large number of channels.

****

Version 1.0.106.1
2016-02-01

****

1) Corrected a minor misformatting in an error message about an incorrectly installed ASIO driver for a device. Also made the message more explicit that the problem was that the ASIO driver DLL could not be opened.

2) You could not set sweep parameters for a signal generator and start or stop the sweep in a single command, you had to use two commands. This is fixed.

3) Added a lot of debug output to show the buffer structures if ASIOCreateBuffers fails. This is to track down an "invalid parameter" failure an ASIO driver is returning for unknown reasons.

4) Added some log info messages to track the timecode status on the first two timecode readers. This it to help track down reports of the timecode readers locking up after about a week.

****

Version 1.0.106.0
2016-01-03

****

1) Fixed two message buffer overrun problems in the code that takes a program dump.

2) If SMPTE Timecode data had many user-data bits set (instead of the default value of all zeros for user data) the timecode reader had trouble locking to the timecode value. This has been corrected.

****

Version 1.0.105.0
2015-12-15

****

1) When running an ASIOGroup with multiple ASIO devices in it, the debug code that checked if the callback was on the same thread as previously would get very confused and make a log message for every callback, slowing things down to a crawl. This has been disabled.

2) Turned off the ASIOGroup debug that displayed a debug message on the first 40 callbacks or so.

3) Removed extraneous newline from "ASIO Driver started successfully" message.

4) Changed the log messages for failures setting thread priority to be more explicit about what the requested priority was.

5) Updated various copyright fields to include 2016.

6) When taking a dump, the date and time in the dump file name are the local date and time, not UCT as in the previous version.

7) There was a slight chance of a crash when loading a preset without first opening the ASIO interface. That has now been covered with an appropriate error message for restoring channel data without an interface. The original patch that covered 99.9% of this case was made in version 82.5 in 02/12/2014.

****

Version 1.0.104.0
2015-11-23

****

1) It looks like Radar is crashing in EnumerateProcessThreads. Wrap it in a try/catch and add some more debugging to tell where it is being called from when and if it crashes.

2) Examined many uses of EXCEPTION_EXECUTE_HANDLER and changed a number of them to try to take a dump when an exception occurs. Maybe this will stop the problem we have with crashing and not taking a dump.

3) Changed a number of TRACE commands to ODS to be sure we get the debug message out even in a release build. These are generally in places where something very bad has gone wrong.

****

Version 1.0.103.0
2015-11-19

****

1) Wrapped ASIO callback in try/catch to catch any possible processing errors. SMS will log a message and take a dump if something goes wrong during callback processing.

2) When the callback thread stops running the overtime messages are sent to the debug output as well as the log. This can make it easier to synchronize the debug log with the main log.

3) When a dump is taken for any reason the notice of the dump is sent to the debug output as well as the log. Again, this helps in seeing major events when only looking at the debug output.

4) Removed unnecessary use of GetThreadId function that is not available on Windows XP Embedded systems.

5) Corrected logging of the license information, which attempted to make two log lines with a single log call. The code now properly makes two single-line log calls.

6) When the ASIO callback thread dies things don't work very well, but if some external source continues to send play commands to SMS, we would get a whole lot of log messages about hung channels, and get a dump every 10 seconds to boot. The dump in this case is pointless, as is waiting 10 seconds before giving up on the play attempt. Shortened the wait interval to 2 seconds in this case, and eliminated the dump. If a playback selection gets hung for some reason while the callback thread is working we will still wait 10 seconds and take a dump.

7) When building an ASIOGroup, if one of the devices had a fixed buffer size, SMS could crash with a divide by zero error.

8) Also, if the device reported any of a number of possible insane values as a result of the ASIOGetBufferSizes call, the bad values would have been ignored and could have caused problems.

9) Some unlikely but possible optimizations were not done when combining devices into an ASIOGroup. These optimizations are now attempted, and if they happen to work they can make a better group blending.

****

Version 1.0.102.0
2015-10-05

****

1) asiosms was issuing a ResetRequest that should have been a Resync Request when it got a sample rate change request when running. This really didn't matter much, since it would only be issued for a sample rate change while running, and we don't allow sample rate changes while running, so the code should never be executed anyway.

2) Changed the ASIO interface to return 0 instead of 1 to a resync request. Maybe that will stop some drivers from issuing a continuous resync request.

3) Fixed a bad debug message that could crash the ASIO driverdurning the ASIOStart routine, without leaving any indications that this had happened.

4) Added a few more debugging messages.

****

Version 1.0.101.0
2015-09-17

****

1) Added additional debug messages to the AsioMultiple driver control path.

****

Version 1.0.100.0
2015-09-16

****

1) Moved project from VC6 to VS2010. Played lots of games with the build file to get things to build with minimal errors, warnings, and general gripes and inappropriate complaints.

2) Fixed some of the priority setting routine to return the correct error code when compiled in debug mode.

3) Added the thread id to several log and debug messages dealing with setting thread priority.

4) Fixed a bug in the priority handling code for Windows 7.

5) Fixed a problem where the main UI thread could inadvertently be set to realtime priority when starting the ASIO driver.

6) Made all debug and log messages mentioning threads consistently display the thread id in hex format to match the debugger display.

****

Version 1.0.86.0
2015-08-15

****

1) Added code to handle a special ASIO driver: "SMS Channel Interface". When installed and configured into SMS, this allows other programs to open SMS as though it was a normal ASIO device. When this is done SMS can open the normal ASIO interface(s), aggregating multiple interfaces into a single interface, and then route data between this combined interface and all other programs that want to use an ASIO interface. As well as routing between the actual ASIO interfaces, SMS can route audio between one program and another.

2) This initial test version of ASIOSMS does not yet have an installer.

   To install the TEST VERSION of ASIOSMS, do the following manual steps:

   1 Install SMS version 1.0.86.0 or later.
   2 Unzip the ASIOSMS zip file into the same directory as SMS. This is usually "C:\Program Files\RSD\SoundMan".
   3 Open a command prompt and navigate to the directory where you unzipped ASIOSMS.
   4 At the command prompt, type "regsvr32 asiosms.dll"
   5 You should get a sound and a popup window that indicates that ASIOSMS was correctly registered.
   6 Close the window and command prompt, ASIOSMS is now installed.

   A full installer will be present in a later version.

3) To use ASIOSMS with SoundMan-Server, you need to do the following steps:

   1 Start SMS before starting other programs that will use ASIO devices.

   2 Check that the SMS Channel Interface is installed correctly.

      Click on the dropdown list in the upper left of the SMS windows that shows the available ASIO devices to select for use. You should see "SMS Channel Interface" as an available selection. DO NOT select this driver at this time! If you do not see "SMS Channel Interface" as an available selection, review the installation instructions above.

   3 Decide which actual ASIO device or devices you wish to use.

      In the SMS dropdown showing the ASIO devices, note down the exact name or the number of the devices you want to use. The first device (listed just under "none") is device 0. The next device is device 1, etc. For instance, you might see a list like:

      "none"
      Interface 0 "ASIO 2.0 - Maya 7.1"
      Interface 1 "ASIO Sample Driver"
      Interface 2 "MOTU FireWire Audio"
      Interface 3 "SMS Channel Interface"

(Only the text in quotes will be showing in the list. You will have to count off the interface numbers your self.).

4 Create the Group ASIO device in SMS.

In the command line for SMS, create a command line that has, all on one line, text similar to the following:

The "ASIO Group" can be any name you want, as long as it is less than 64 characters. This will be the name of a 'group device' that you will open in SMS. It will only be visible in SMS, not in other programs that open ASIO devices.

In place of "MOTU FireWire Audio" select the name of the first real ASIO device that you wish to use. You can also use the device number (2 in this case) in place of the quoted string. (See the documentation on CONFIG SET ASIOGROUP in the SMS documentation.)

If you want to use more than one real ASIO interface, you can put multiple "interface" lines in the ASIOGROUP command. All of them must occur BEFORE the "SMS Channel Interface" entry, which MUST BE THE LAST ENTRY.

```
config set asiogroup "ASIO Group"
        interface "MOTU FireWire Audio"
        interface "SMS Channel Interface"
```

Once you have typed the entire command line, hit the Enter key to pass the command to SMS. There should be no response on the "Response" line just below the command line if everything was formatted correctly and was accepted by SMS.

5 Open in SMS the group device just created.

Click on the ASIO device dropdown list in the upper left corner of the SMS interface. You should now see a new ASIO device available named "ASIO Group", or whatever you named your ASIO group device.

Select this device from the dropdown and open it with the Connect button. If all is working correctly, in a few seconds you should see a number of VU meters appear at the bottom of the SMS window. There will be one VU meter for each real output channel on the real hardware interface(s), and then a number of VU meters of the output channels to the SMS ASIO interface. The SMS ASIO device will have the same number of input channels as the real hardware device has outputs, and will have the same number of output channels as the hardware device has inputs. This may seem confusing at the moment, but there is a reason for doing it this way, which will become clear momentarily.

6 Route the audio in SMS.

The real ASIO devices and the SMS Channel Interface are open and running, but no audio is yet routed, so nothing will be heard. You need to decide how you want the audio routed before you can complete this step.

Here we will route all of the real ASIO inputs to the SMS Channel Interface outputs, and all of the SMS Channel Interface inputs to the real audio outputs. The result of this will be that any other program opening the SMS Channel Interface ASIO device will have access to all of the real ASIO inputs and outputs.

Assume that the real ASIO devices collectively have 32 inputs and 32 outputs. Since we put the real ASIO devices first in the ASIOGROUP, the real audio channels are inputs 0 to 31 and outputs 0 to 31. The SMS Channel Interface inputs and outputs are channels 32-63.

First, let us set all of the inputs and outputs to full gain:

SET CHAN I0-63 O0-63 GAINDB 0

Now we can deal with the crosspoints.

We want the real hardware inputs routed 1:1 to the SMS Channel Interface outputs. This would be input 0 to output 32, input 1 to output 33, etc. We can do this by setting a diagonal, again to full gain:

SET CHAN DIAGONAL X0-31.32 GAINDB 0

This makes all of the real hardware inputs available as inputs to anyone opening the SMS Channel Interface.

Now we want to route all of the SMS device inputs to the real hardware outputs. This will route the outputs of any programs using the SMS Channel Interface to the real hardware outputs:

SET CHAN DIAGONAL X32-63.0 GAINDB 0

7 Setup is complete. Test some other ASIO-using program with the SMS Channel Interface to see that it has access to all of the real ASIO inputs and outputs.

Note: If you have multiple hardware interfaces they will be running at nominally the same sample rate. It is YOUR RESPOSIBILITY to see that they are sample-locked in some manner or other, such as using word clock sync with a common sync generator for all devices. If you do not synchronize the devices you can expect clicks and pops from all interfaces.

8 Simplify future setup by creating a script file to do the work.

In the My Documents folder you will find a folder called SoundMan Scripts. In this folder create a text file with Notepad or some similar text editor. Do not use Word or some other formatted-document program as the resulting text lines will probably be damaged by the automatic formatting.

For demonstration, create a file named "STARTUP.TXT".

Start by entering two lines in the text file:

SCRIPT STARTUP BEGIN

```
SCRIPT STARTUP END
```

Find the CONFIG SET ASIOGROUP line that you created for SMS. You can do this by looking back in the SMS history window if you did not write it down when you created it.

Enter this ASIOGROUP line in the script file just after the SCRIPT STARTUP BEGIN line. You should have:

```
SCRIPT STARTUP BEGIN
config set asiogroup "ASIO Group" interface "MOTU FireWire Audio" interface "SMS Channel Interface"
SCRIPT STARTUP END
```

Following this line, enter a line to open the new device you just created. This is the equivalent of selecting the device from the dropdown list and clicking the CONNECT button:

```
CONFIG SET INTERFACE "ASIO group"
```

Now enter the lines to do the audio routing. For our example we should now have:

```
SCRIPT STARTUP BEGIN
config set asiogroup "ASIO Group" interface "MOTU FireWire Audio" interface "SMS Channel Interface"
CONFIG SET INTERFACE "ASIO Group"
SET CHAN I0-63 O0-63 GAINDB 0
SET CHAN DIAGONAL X0-31.32 GAINDB 0
SET CHAN DIAGONAL X32-63.0 GAINDB 0
SCRIPT STARTUP END
```

Now save this file.

The next time SMS is run, you can enter on the command line:

```
START SCRIPTFILE STARTUP
```

This will do all of the necessary setup.

****

Version 1.0.85.0
2015-02-07

****

1) It was possible to crash a thread creating a log record in a low memory condition. This caused things to pretty quickly go downhill. The log write is now protected against most possible errors. A new warning message may appear in the log if messages are lost as a result of the log write failure.

2) Updated copyrights to 2015.

****

Version 1.0.84.0
2014-09-17

****

1) The first log record could show an incorrect time value.

2) Crosspoint delay time setting was unreliable. Sometimes you could get the requested delay time set and sometimes it would be ignored.

3) Input and output delay setting was not working after a patch in the last release to set the values synchronously to make sure the correct delay time was set in all cases.

****

Version 1.0.83.0
2014-05-19

****

1) The performance of input and output conversions between 32 bit integers and floating point has been improved. This is important for newer ASIO drivers that treat audio as 32 bit values rather than 16 bit values.

2) SET MATRIX could take a very long time on a large matrix with hundreds of input and output channels. This has been drastically improved.

3) The internal log is cleared every 65000 messages. If SoundMan-Monitor is running and messages are coming quickly, there was a chance that log messages just before the internal log was cleared could have been lost. There is now a short delay of a half second before clearing the internal log to give SMM a chance to grab the current values.

4) When clearing the log, the last 100 or so messages of the old log are saved and put into the cleared log that is maintained in memory. This in-memory log is included in any crashdumps created by SMS.

5) Previously if many messages to log occurred at once, and some of them were queued for logging while others were made directly, the messages would be logged out of order. Logging has been redone so that all messages are now logged in correct order, even when they occur in a quick burst.

6) A few messages still identified channel solely by channel number rather than channel name, making it hard to decode what the channel really was. These have been updated to use the channel name.

7) Disabled a couple of debug messages installed in version 82, since they were of little use and generated huge amounts of noise in the log files.

8) The startup interface between SoundMan-Assistant and SoundMan-Server has been improved. In the past SMA could start two copies of SMS when trying to get the system started. That should now be a thing of the past. Also SMA could start running and send the show initialization cues before SMS was ready to receive them. This should also be a thing of the past.

9) When SMS is started from the Startup menu on a machine that has just finished booting, it can take anywhere from 6 to 20 or more seconds to display the tray notification icon. In the past SMS startup waited for this to occur, which greatly delayed startup in some cases. Now we run the tray notification on a separate thread, so the startup will not be delayed.

10) Eliminated a debug message about setting the current delay point on channels.

****

Version 1.0.82.7
2014-02-21

****

1) If a script fails to start it was not obvious why. Now the error message will show why the start failed.

2) If a script encounters an error the script is blocked with the error flag set until it is cleared. It isn't always obvious how you got into this state, so a new warning message indicates that the script has been blocked when the error occurs.

3) If a script got an error on a normal SMS command the error message would have been logged twice. Also, GET message responses from a script would have been logged, even though they are normally not logged. These are both fixed.

4) Previously if a script got an error on any command it was marked as "error seen" and blocked from further execution. The only way this could be cleared was to reload the script. Since this is awkward when not debugging a script, the script will now be marked as stopped when it gets the error, and it can be run again normally. Also, a warning message will indicate that the script stopped as the result of the error, which is typically logged just before the warning message.

5) Removed the old POST_IT method of logging certain log messages in time-critical paths, and replaced the calls with the much better PostLogMessage routine calls. This should generally be transparent to users, but in a few cases it might result in small changes in the log message order.

6) SMS was allocating twice as many delay buffers as were actually required when setting the delay for a crosspoint. This is now fixed.

7) Setting the crosspoint delay could sometimes result in a delay value that was off by one buffer. Appropriate interlocking has been added to see that this can no longer happen.

8) If more than three rows of VU meters had to be displayed the result was somewhat ugly when the meters were showing a signal indication. This is fixed.

****

Version 1.0.82.6
2014-02-16

****

1) If you tried to create a script or load a scriptfile with the ASIO interface closed, SMS would crash. This is fixed.

2) A few error messages associated with script file parsing were incorrectly formatted, with an extraneous colon following the word ERROR. These have been corrected.

3) The "Log file saved" log message did not have an "I:" information flag on the front, as it should have. This resulted in a mis-formatted log line.

4) Added a call to SetErrorMode to prevent the stupid "what do you want to do" message popup that sometimes shows up if SMS crashes. Now SMS will just take the dump rather than hanging.

5) Found a problem with restoring EQ info that could cause a crash. This is now fixed

6) If we have to allocate a new EQ section for a channel when restoring the EQ info and we are unable to create the needed EQ section, we will abort the preset load. There will also be a Warning message indicating the name of the channel and the EQ section that could not be created.

****

Version 1.0.82.5
2014-02-12

****

1) Attempting to load group info from a preset without first opening the ASIO interface would crash. The error is now detected and an error message will result from the preset restore attempt.

2) VU meter layout could crash with a large number of VU meters, apparently from running out of system resources. There is now code that protects against this sort of thing causing failures.

3) Channel delays were not being restored correctly from a preset. The values for the delay and enable were restored, but the actual delay line didn't get set to the correct position.

4) A debug Message Box call was erroneously left in the last version in an error path. This resulted in SMS hanging on headless systems if the error condition occurred. The message box call has been removed, the error handling code improved, and locking added to hopefully prevent the error condition occurring in the first place.

5) A deadlock situation that resulted in playback hangs on version 82.0 has been resolved.

6) A script can now run another script.

7) A crash could occur if you specified zero playback channels when initializing the ASIO interface. This is fixed.

****

Version 1.0.82.4
2013-12-04

****

1) Fixed a divide by zero in the last version when trying to make a log message that complained about a driver misconfiguration.

2) Added some more warning messages about possible driver misconfigurations.

****

Version 1.0.82.3
2013-12-04

****

1) Clarified some of the previous debug messages.

2) The mainline ASIO open code was erroneously checking that the preferred buffer size was a power of 2 if the granularity was 0. It should have been making this check if the granularity was -1, not 0.

3) Modified the mainline open code to try to open an device with a preferred size that is not binary (when the device said it should be by setting the granularity to -1), but the preferred size is a multiple of the minimum buffer size.

4) Added some warning messages if on a binary buffer size device the preferred size is not a power of 2, and whether it is or not a multiple of the minimum buffer size.

5) The AsioMultiple CreateBuffers routine was also checking that the preferred size needed to be binary when the driver required power of 2 buffer sizes. This failed if the driver supplied a preferred size that was not a power of two. Changed the code to only complain if the preferred size requested did not match the size provided by the driver.

****

Version 1.0.82.2
2013-12-04

****

1) Add mode debug log messages in the AsioMultiple buffer setup, and fixed errors in some of the converted messages from the last version.

2) Added a special path in AsioMultiple to handle drivers that can violate the specification to create a compatible buffer size. A driver that specifies that its buffer must be a power of 2 should not allow a preferred size that is not a multiple of 2. But some drivers can do this. The normal search for matching buffer sizes will fail, but a final last-ditch check can notice that all drivers have been manually set to a compatible buffer size and use that size.

****

Version 1.0.82.1
2013-12-03

****

1) Converted a bunch of TRACE messages into log messages to give some better idea of the inner workings of trying to match multiple ASIO devices in AsioMultiple.

****

Version 1.0.82.0
2013-11-11

****

1) Cleaned up an extra carriage return on the end of an output debug message that comes out when several commands for the same channel are entered very quickly. Also adjusted the debug code so that the message is much less likely to appear in normal operation.

2) Dump files are now better formatted internally, and easier to analyze.

3) The DIAGONAL syntax added in version 1.0.79.0 was not clearly documented. This adds the word DIAG or DIAGONAL preceding a crosspoint range specification in a command. When the DIAGONAL keyword appears, only the diagonal crosspoints in the range will be set.

   For example:

   SET CHAN DIAGONAL PX0.0-5.5 GAINDB -10

   This will set the gain for channels PX0.0 PX1.1 PX2.2 PX3.3 PX4.4 and PX5.5 to -10 dB. The rest of the input and output channels in the range of 0-5 will not be affected in any way.

4) Scripts were not waiting correctly when waiting for time, and would end up using processor time unnecessarily. This made many scripts unusable on single processor machines. This is now fixed.

****

Version 1.0.81.7
2013-09-24

****

1) Changed the message indicating that the interface refused to run at the requested sample rate from an information message to a warning message, since the user didn't get what they wanted, and this may be important.

2) Added a bunch of debug messages in the preset store path to try to determine why SMS can't store a preset. Probably running out of memory.

3) Modified the preset memory allocation routine used when saving a preset to not be as aggressive on requesting memory space for the preset data buffer. This should make it more likely a preset will store in a low memory condition with a large interface.

4) Modified the working set reservation code to try to reserve up to 3 GB for SMS, rather than the previous limit of 1GB. This may help reduce paging when running a large interface.

5) Saved presets are now smaller when a large matrix is saved. This is done by saving the parametric EQ info in a new more compact format. This version of SMS can load the eq info in both the old and new formats, but only saves it in the new format.

   This means that presets saved with this version are NOT backward compatible to older versions of SMS!

6) When there are enough channels displayed that the scroll bar appears for the VU meters, the meters did not always display correctly after a scroll. Channels with no output would show the output of the channel that had previously been displayed on the same meter before the scroll, rather than showing no output as it should have. This is now fixed.

****

Version 1.0.81.6
2013-08-26 SCS only

****

1) Added some SCS key debugging messages.

****

Version 1.0.81.5
2013-08-25 Steve only

****

1) The fix in the last version to not complain when relocking the command line with the same key did not work in all cases. This fix should be better.

2) Improved the flagging in the log messages for commands. They should now be easier to sort out from responses and informational messages.

3) Fixed a copyright date in the About box.

4) Added log messages to preset loading to help diagnose load failures.

5) Storing a preset that requested all channel data would fail. This is fixed.

6) Restoring the connection info from a preset did not work correctly.

7) The Ecoustic branded version of SMS now defaults channel delays to disabled. The non-OEM version defaults to enabled for compatibility with the AB64.

8) Fixed a copyright date on the splash screen.

****

Version 1.0.81.0
2013-08-14 Waterworld Only

****

1) Improved log messages on setting process working set size.

2) When SMS gets an invalid command it now logs the error with the invalid command word and the entire invalid command line. This can help diagnose TCPIP errors.

3) Added CONFIG SET TCPDEBUG { YES | NO } command and CONFIG GET TCPDEBUG command. Setting TCP debug makes extra log messages that can help diagnose link problems with the command generator program.

4) Trying to lock the command line a second time using the same lock key would return an error message that the command line was already locked. Now the error message will only occur if you try to lock a locked command line using a different key. The second lock attempt will fail and the command line will remain locked under the original key.

5) There was a bug in the TCP command parsing that could under rare occasions cause a command to be received incorrectly.

****

Version 1.0.80.1
2013-08-10

****

1) Some excess debugging code that should have been removed before release was accidentally left in the previous version. It is now disabled.

2) Recent changes to speed up channel name lookup caused group names to not be found. This is now fixed.

3) The attempt to set the process working set size to 1GB could fail on smaller computers running a 32 bit OS. SMS now tries 1/2 and 1/4 GB sizes if it cannot reserve 1GB of working set space.

4) In CommandCue mode port names could not be referenced by name due to recent changes. This also is fixed.

5) Some multiprocessor debug code was inadvertently left active that could cause the "Force single threading" checkbox on the interface to flicker.

****

Version 1.0.80.0
2013-08-07

****

1) The Set Interface parsing code had an error that prevented operation with an SCS-type dongle. This is fixed.

2) The VU meter display has been reworked to make the meters more usable with a large number of channels.

3) In some cases when trying to open an ASIO interface that was not available (for instance, the Firewire cable was unplugged) SMS could crash after the failed open. This has been fixed.

4) The log message that indicates that SMS could not load the startup preset is now flagged as Informational rather than Warning, as most users do not have a startup preset.

5) When using RME audio cards, their TotalMix mixer is started by default. On smaller configurations and on Windows XP this is usually not a problem. On larger configurations, especially on Windows 7 and beyond, having TotalMix running can result in occasional audio clicks.

   If you are running RME hardware and have clicks, we recommend that you turn off TotalMix.

****

Version 1.0.79.0
Never Released

****

1) Added a new SCS operating mode to SoundMan Server to allow SMS to work with dongles that are keyed for SCS control only. An SCS dongle will only allow SMS to be opened by SCS.

2) Code has been added to set realtime priority on Windows 7 and later OS versions. This should help eliminate the occasional click that might be present on the newer OS versions, which changed the method used to set thread priority.

3) Setting the preset path in one invocation of SMS was not being found in time to load the initial preset from the new path the next time SMS started.

4) Loading a preset from a system with fewer channels into a larger system would end up looping trying to load the preset. This is fixed.

5) When loading a script file there is now a "Script file X loaded" information message displayed.

6) When restoring a preset there is now a "Preset X restored" message displayed.

7) Channel and crosspoint name lookup has been sped up by a factor of as much as 100 times over the previous code. This is most noticeable when running a script that references lots of crosspoints and channels.

8) Memory allocation and deallocation has been sped up. Previously it could take up to several minutes to close the ASIO interface on very large channel configurations.

9) Typing the name of a script file incorrectly could cause SoundMan to fail.

10) Negative numbers in scripts did not work well.

11) Input and output channels were not being correctly restored from a preset in the latest versions of SMS.

12) Exiting SMS by clicking the Exit menu item while the interface was open and scripts were loaded would crash. This is fixed.

13) Assigning a value to a variable in a script on the VAR command (rather than the ASSIGN command) did not actually do the variable assignment, the variable remained with a value of zero.

14) The script IF command has been enhanced. As well as being able to GOTO a label, you can now directly do an ASSIGN or LET statement as the result of the condition being true.

    The two forms of IF statement are now:

IF CHAN single-channel
    testable-item relation fixed-value
    GOTO labelstring

IF CHAN single-channel
    testable-item relation fixed-value
    THEN ASSIGN varname = expression | CHAN single-chan testable-item

15) Corrected some minor alignment problems with windows on the main SMS interface.

16) Scripts now allow variable substitutions inside of strings, when the string is used as a form of macro for use in subsequent commands. For instance the following script fragment is now valid:

```
var chantype = "X"
var inchan = 0
var outchan = 0
var chanid = "chan @chantype @inchan.@outchan"
set @chanid gaindb -10
```

17) You can now chain multiple text concatenations in a single assignment command in a script. This makes it considerably less verbose to build strings or commands that contain multiple items that must be substituted.

18) SMS is now "large address aware", which lets a 32 bit program use up to 4GB of memory. Without this option set in the build, a 32 bit program can only use 2GB of total memory. This helps with larger configurations which might otherwise run out of memory.

19) If SMS runs out of memory when trying to open the ASIO device (which can only happen with very large channel configurations) you will now get a clear error message about the problem. Previously the interface would just fail to open for no obvious reason.

20) A potential startup crash that would occur on Win7 has been fixed.

21) Added the ability to load and run a script in a single command.

As well as the previously existing

    RUN <script name>

command, you can now say

    RUN SCRIPTFILE <script file path>

The first form will load a script that has been previously loaded. The second form combines two commands:

```
SCRIPTFILE <script file path>
RUN <script name found in script file>
```

The new format means that you can not have a script named SCRIPTFILE, but that is a good thing, not a restriction.

22) If a script file failed to load correctly (usually due to an internal malfunction) it was difficult to determine why, as the error message indicated the wrong error in most cases. This has been improved.

23) A crosspoint channel range specification can now be preceded by the word DIAG or DIAGONAL. This will indicate that only the diagonal crosspoints should be affected by the command. "Diagonal" crosspoints are those crosspoints where both the input and output channel number increments between each crosspoint, for instance, x0.0, x1.1, and x2.2 are diagonal crosspoints, but x0.0 x1.0 x2.0 are not.

When setting a diagonal set of crosspoints you must give the same number of input and output channels, for instance "x0-3.0-3". Alternately, if the diagonal is something like "x18-27.43-52", it can be hard to mentally come up with the final channel number for the outputs if you only know the input channel range of interest. To make this easier to deal with, you can just give the input range or only the output range: "x0-3.5" or "px20.0-7".

24) If a command line contained tab characters, they would appear as ugly black rectangles in the log. They now appear as spaces.

25) The EQ band parameters displayed in response to setting an eq band were displayed with zero-relative band numbers. They are now correctly displayed with one-relative band numbers.

****

Version 1.0.78.0
2013-03-07

****

1) Replaced the logging timer routine with a more accurate version.

****

Version 1.0.75.0
2012-05-29

****

1) An error in the previous version would cause some files to not play to timecode.

2) Group info was not being correctly loaded and saved from presets. This is now fixed.

3) Added the command CONFIG GET MACHINEID. This will return a response in the format:

MachineId = XXXXXXXX

Where XXXXXXXX is the 8 digit machine serial number for the machine that SMS is running on. The machine id is unique to a particular machine, and will be present even if no dongle is present.

This command is similar to CONFIG GET SERIALNUMBER, which gets the 8 digit serial number for the dongle, if one is present.

4) Added two predefined variables that can be accessed from scripting. "MachineId" returns the unique machine id for the current machine. "MachineSerial" returns the serial number of the dongle if a dongle is present. Both of these values are constants, so cannot be changed by the script, only interrogated.

5) Graphics EQ on a crosspoint channel would not work unless the parametric EQ for that crosspoint channel was also enabled. This was incorrect, as graphic and parametric EQ enables are supposed to be independent of each other.

6) Turning parametric EQ on or off inadvertently would also turn graphic EQ on or off at the same time.

7) Setting the FFT quality for the graphic eq produced an error.

8) FFT work buffer allocation was assuming a single processing thread, which was incorrect. Work buffers are now allocated for all possible processors.

9) Two new channel GET commands have been added. These are similar to GETVU, but instead of getting the current VU value as a range of 1 to 128 that is suitable for a VU meter display, they return the channel level either as a floating point 0..1 number, or as a value in dB.

```
GET CHANNEL <channels> LEVEL          # returns Level          chan=fpt
GET CHANNEL <channels> LEVELDB        # returns LeveldB        chan=fpt
```

10) The syntax to set pitch shifter parameters has been improved.

```
//      SET CHAN|CHANNEL channel-spec
//              PITCH
//                      ON | OFF
```

```
//                  [SHIFT] fpt                         in semitones (12 = 1 octave!)
//                          FADETIME fpt
//                          FADETC          tc
//                  QUAL | QUALITY          n                       1..8
//                  FFTSIZE                 n           128..8192, powers of 2 only
//                  OVLP | OVERLAP          n                       1..32
```

Previously you could say "SET CHANNEL <chan-list> PITCH <value>" to set the pitch shift in semitones. This is still possible, but you can now be more explicit and say "SET CHAN <channel-list> PITCH SHIFT <value>".

Also, you can now set the overall quality of the sound of the pitch shifter easily using the same QUALITY parameter that was available for the graphic equalizer (which is also implemented using an FFT). This is simpler than having to set the FFT size in samples and the overlap factor.

The QUALITY value will set both FFTSIZE and OVERLAP at the same time. The values are:

QUALITY	FFTSIZE	OVERLAP
1	512	4
2	512	8
3	1024	4
4	1024	8
5	2048	4
6	2048	8
7	4096	4
8	4096	8

While the overlap and FFTSize values selectable with QUALITY will give a good coverage of the typically usable values, it is possible to manually set values outside the range and combinations possible using the QUALITY setting. In some select cases it may be possible to get better sound or possibly better sound at a lower processor usage by using the manual settings, however experimentation will be required. Experimentation is still required using the QUALITY setting, but it is simpler.

As the quality increases the processing overhead also increases, quite drastically. The lowest acceptable quality should be selected if more than one channel is going to be pitch shifting.

It is possible to run out of available processor power with a smaller machine or with a number of pitch shifters and graphic EQs enabled. This will have the usual bad effect of audio dropouts, crackling, and other bad things. If this happens the only solution is to reduce the quality or number of enabled graphic EQs and pitch shifters.

11) The graphic EQ and pitch shifter commands referring to the QUALITY parameter can now be abbreviated as QUAL. This affects the following commands:

```
SET CHAN <chans> GEQ   QUALITY n
SET CHAN <chans> PITCH QUALITY n
GET CHAN <chan>  GEQ   QUALITY
GET CHAN <chan>  PITCH QUALITY
```

All of these commands can now be abbreviated as:

SET CHAN <chans> GEQ QUAL n
SET CHAN <chans> PITCH QUAL n
GET CHAN <chan> GEQ QUAL
GET CHAN <chan> PITCH QUAL

12) Getting pitch shifter status values did not work correctly.

13) Scripts loaded with the SCRIPTFILE command now have a default directory location, so it is no longer necessary to use a full path. The scripts can be stored in the "SoundMan Scripts" directory in My Documents.

14) The graphic EQ processor had a bump at 22KHz if all of the bands were taken down to a lower gain. This is because 22KHz would be band 32, which does not exist as a settable frequency band, so was remaining at 0dB gain even if band 31 was set to some other level. The frequencies above band 31 are now included at whatever value band 31 is set to.

15) The available scripts in the SoundMan Scripts directory can be listed with the new command

CONFIG GET SCRIPTFILES

This command returns a multi-line response ending with a line containing a single ".". The first line is "ScriptFiles:". Each subsequent line will contain the file name for a script file in the "SoundMan Scripts" directory. The final line has the single period on it. If there are no script files in the directory the response will consist of the header line and the final dot line.

16) The currently loaded and available scripts that can be run can be listed with the command

CONFIG GET SCRIPTS

Previously the command LIST SCRIPTS would do this. This command was temporary for development of the script facility and has been removed.

This command returns a multi-line response ending with a line containing a single ".". The first line is "Scripts:". Each subsequent line will contain the name and status of a script that is currently in memory. The script may be running, waiting for some event, stopped, or have an error of some kind. Each line will contain the name of the script followed by a colon, and then the status of that script.

The status part is multiple words and is human readable. For the convenience of controlling programs, the first word of the status, immediately after the colon, is a 3 digit status number.

The number of status values may change over time. While it is unlikely that any current status values will be deleted, more status values may be added in the future. The first digit of the status value indicates the general kind of the script status, and the remaining two digits provide more detail.

The current status number and values are:

000 IS INVALID.
001 HAS AN ERROR.
100 is stopped.
200 is running.
300 is waiting for time.
310 is waiting for timecode.
320 is waiting for an event to happen.

The final line of the response has the single period on it. If there are no scripts in memory the response will consist of the header line and the final dot line.

15) The available presets in the SoundMan Presets directory (or whatever the current preset directory is named) can be listed with the new command

CONFIG GET PRESETS

This command returns a multi-line response ending with a line containing a single ".". The first line is "PresetFiles:". Each subsequent line will contain the file name for a preset file in the current preset file path. The final line has the single period on it. If there are no preset files in the directory the response will consist of the header line and the final dot line.

The current preset path can be determine with the command

PRESET GET PATH

This will return a single line response in the following format:

PresetPath="path name"

The preset path can be changed from the default or current value with the command:

PRESET SET PATH "path name"

It is recommended that to reduce confusion in show setup that the preset path NOT be changed unless it is absolutely necessary.

16) If the license info in the dongle used the full available string size, as occasionally happened on International orders, the splash screen formatting was very poor, leaving most of the lines trimmed at both edges. The splash screen has been widened and the formatting adjusted so that the splash screen looks good with any amount of license text.

****

Version 1.0.74.0
2012-02-23

****

1) Removed a bunch of extraneous newlines from the end of log messages added in the last few releases.

****

Version 1.0.73.3
2012-01-08

****

1) If no file is loaded on a channel the sample rate now defaults to 48K instead of 0.

2) Setting a stop point on a channel with no file loaded could in some cases cause SMS to crash.

****

Version 1.0.73.2
2012-01-08

****

1) There was a possible deadlock situation if a channel, which was part of a channel group attached to a time code reader, received a second Play command just as the time code started moving and the time code jumped. Getting this lockup required sending two Play commands, one before time code started to move, and the other just as the time code jumped from one value to another fairly far apart. This is now hopefully fixed.

****

Version 1.0.73.0
2011-12-31

****

1) All preset commands required an open interface. Only a PRESET SAVE command should have required an open interface. This has been fixed.

2) Under rare conditions SMS could crash when starting to play a group of playback channels, due to a timing window. There is a temporary fix to prevent the crash, and also take a dump to better understand what is causing the timing window.

3) Log messages have been added to show when timecode starts and stops on a timecode reader.

4) Debug log messages have been added to show when playback pauses or resumes because timecode starts or stops on tracks that are locked to timecode.

5) The periodic log message that shows the number of playing tracks and system utilization has been modified to show the number of playing tracks that are paused because they are locked to timecode and the timecode is stopped.

6) When saving a preset for the first time you would get a warning message saying "Unable to rename preset". This was a harmless message that only indicated there was no old preset to rename, but it should not have appeared under those conditions.

7) If a script file had errors in it, there was a small memory leak.

8) The timestamp for log records could in some cases be off by several minutes from the correct value. This has been corrected.

9) Changed the copyright to include 2012.

****

Version 1.0.72.5
2011-12-04

****

1) When attempting to create an ASIOGROUP with a device that is broken or the hardware not available, the ASIO driver error was returned directly as a command response. The driver response should have been preceded by the standard "ERROR <number> - " text formatting so that the caller could recognize that a failure of some sort had occurred.

   The ASIO driver error text is now preceded by "ERROR 0100 - " to flag it as an error response.

2) The text in dongle serial number field in the main window can now be selected with the mouse and the text copied to the clipboard.

3) Script command processing no longer erroneously returns an "OK" response at the end of the script. This prevented one script from being called from another script.

****

Version 1.0.72.4
2011-12-01

****

1) Clearing the resume and stop events to break out of a loop could end up causing SMS to lock up. This is fixed.

2) The "force single threading" checkbox will always be set on a single processor system. Previously it could be cleared manually, but this didn't actually have any effect.

3) When configuring an ASIOGROUP you need the word INTERFACE before the name of each interface to use in the group. Previously if the second or later word INTERFACE was missing the group would be created with only the initial interfaces, and no error would appear about the unrecognized text following the first interface name.

4) On a multiprocessor system, SMS could lock up while exiting.

5) The always-confusing matrix mode radio buttons have been removed from the interface window and replaced with a hopefully slightly less confusing display of the current operating mode.

6) The time values in the Syncstart and Syncstop log messages were wrong when the file sample rate was different from the interface sample rate.

7) Changes in single-processor processing mode are now logged.

8) The STOP, PLAY, PAUSE, and RESUME commands are now synchronous. Previously these commands would return a result to the requestor as soon as the action was started. However it could take some amount of time for the command to actually complete, as it might have to wait for files to be ready to play or the like. Between the time the command completed and the action was complete a new command could be issued, for instance another STOP or PLAY for the same group of channels. This could result in internal confusion as the play/stop state changed unexpectedly.

   The command will now wait until the requested action is complete before returning to the requestor. Also, if the action takes more than one second to complete, log messages will be made every second while waiting. If the action takes more than 10 seconds a dump will be taken, as this should not happen.

****

1.0.71.0
2011-09-08
General Release

****

1) Some very minor cleanup of changes in version 70.7.

2) Removed some unnecessary debugging code.

3) Considerably more debugging messages will be noticed in the log since the last release of SMS. Other than taking a bit more log space these should have no effect.

****

Version 1.0.70.7
2011-07-20
USF Dervish

****

1) Added a monitor to the ASIO callback thread to output a debug message if it hangs up and stops running.

2) Added debug messages to show if a selection has hung around after it should have been deleted which process owned the selection.

****

Version 1.0.70.6
2011-07-15
USF Dervish

****

1) When reading the timecode value from the timecode readers, they return the time in the current timecode format. There is a new option to always get a 30 fps timecode value from the reader, regardless of the current framerate: GET TCREADER <n> TC30. This will return a single-line response in the form "TC30=<timecode> ;".

2) You can now read the time from the timecode generator as a timecode value as well as a time in seconds. GET TCGEN <n> TC. This will return a single-line response in the form "TC=<timecode> ;".

3) When reading the timecode value from the timecode generators using GET TCGEN TC, they return the time in the current timecode format. There is a new option to always get a 30 fps timecode value from the generator, regardless of the current framerate: GET TCGEN <n> TC30. This will return a single-line response in the form "TC30=<timecode> ;".

4) The log time can drift considerably from reality on the Dervish boxes. This should now be fixed.

5) Added debug log messages for an unusual condition that can happen if too many commands are received in a short time.

****

Version 1.0.70.5
2011-07-11
USF Dervish

****

1) Removed the message box that showed up when trying to start SMS a second time, and replaced it with a tray message. This prevents accidentally started copies of SMS from hanging around forever on unattended systems.

2) The VU meters on the output channels were incorrectly pre output delay, so tended to make you believe that the output delay adjustment had no effect.

3) The GET TCREADER <chan> SHAPE command now correctly returns the code type being read.

4) The GET TCREADER <chan> RUNNING command now properly returns YES when the timecode is detected and is incrementing. Previously it always returned NO.

5) Removed some message logging overhead in the playback start path that could sometimes result in a slight delay in the playback start.

****

Version 1.0.70.4
2011-06-30
USF Dervish

****

1) Increased the timeout on trying to create the tray icon back to 60 seconds to prevent starting without a tray icon, which is common currently.

****

Version 1.0.70.3
2011-04-18
USF Dervish

****

1) An assortment of problems with input and crosspoint delays that could lead to random non-zero delay values have been fixed, finally.

2) Most of the Preset code from the previous stream merged back into this stream.

3) Some additional warning and debug log messages added to keep track of system operation in bug reports.

****

Version 1.0.70.2
2011-04-04

****

1) Fixed a crash in setting up group channels.

2) Fixed a crash in exiting the program with a group channel set up.

3) Fixed some formatting problems with the main window.

****

Version 1.0.70.1
2011-04-02

****

1) This is a slightly updated version of version 61 to see if it works better with keeping tracks in sync and not jumping around on track starts.

   Note! Since this version was originally made off of version 61, a great deal of code from many succeeding versions has been added, so this is now a version at least around version 68 or 69.1 as a basis, with some code even following that. The comments for the versions up to version 68 have been added back in below, as they are probably largely applicable to the 70.5 or so version, and we have no notes as to when the code was merged in.

2) Moved the dongle and machine id to the top of the dialog to make them easier to find.

****

Version 1.0.69
2011-03-19

****

1) On a dual processor system the "force single threading" checkbox was checked erroneously.

2) The size of the TCP buffers to the control port has been increased to 65K bytes.

3) Improved the log messages for TCP send errors.

4) Changed the year from 2 to 4 digits in the log records.

5) Fixed a very unusual crash that could occur with many short tracks playing very frequently.

6) Made additional improvements in the TCP/IP command handling path. Under rare conditions a message could get "lost" in the input buffer and not cause an event when it arrived. We now check periodically to see if anything is sitting and waiting.

7) Removed extraneous newline from the end of the dongle id message.

8) Removed extraneous double-spacing in the "Licensed for" message.

9) Changing the stop point on a track while playing now works under all conditions.

10) Previously a command of the form:
SET CHAN P0 TRACK FILE "C:\MySound.WAV" START 30 STOP 30 REPEAT 0
when the file was shorter than 30 seconds, would cause SMS to loop.
This has been fixed.

11) Fixed a rounding error in the TimeToTimeCode routine that could result in loss of frame accuracy when times over 5 hours were used.

12) When creating an ASIOGROUP with a single device, sometimes the command would incorrectly fail with "Incompatible driver buffer sizes." This is an impossible occurrence when only one driver is in the group.

****

Version 1.0.68
2011-02-28

****

1) Showing the ASIO control panel is now logged.

2) Setting or clearing the "single threading" checkbox is now logged.

3) The detailed callback logging number format has been improved.

4) Added a hack to the distributed callback processing to leave one processor unused on a multi-processor machine. This will help in some cases where other things need to run on the system at the same time. It may also hurt, if a lot of callback processing time is required, depending on the design of the ASIO driver. (RME drivers require this change.)

5) If an RME driver is detected the callback thread is reaffinitized to the first 4 processor cores rather than one single processor core as RME set it up. This increases total processing power by about 20% on test systems.

6) It was possible to get the playback code into a loop processing the next event in sequence, due to floating point roundoff errors. This has been fixed.

7) It is now possible to set the channel delay value in samples using the syntax:

    SET CHAN <channels> DELAY SAMPLES <number>

    Previously (and this will still work) you had to say

    SET CHAN <channels> DELAY TIME <number> SAMPLES

8) In all commands that use the word SAMPLES, you can now also use SAMPLE.

9) The Nagle algorithm has been disabled on the TCP/IP control ports. This could at times result in undesired command slowdown and the appearance of a blocked input port. Data flow should remain smooth now.

10) The TCP/IP close logic has been changed to keep a closed port from hanging for long periods of time, or possibly forever. This makes it much more likely that input ports closed due to an error will be recovered. Under very rare conditions it used to be possible for SMS to run out of input control ports. This should no longer be the case, or at least be very much less likely to ever happen.

11) The callback processing threads are no longer affinitized to specific processors. Instead they can float around. This makes them work better with ASIO drivers that have a thread that is nailed to a specific processor and which does a considerable amount of work.

12) If you set the start, resume, and stop points for a track, and then modified

the start point, it erroneously copied the start point value into the resume point, so the loop would loop back to the start point and not the originally intended point.

13) If a track is set up with a resume point beyond the stop point, the track will play from the start point (or start of track, if no start point is set) to the stop point. It will then skip to the resume point, and then loop from the resume point to the end of the track until stopped.

Previously setting the resume point past the stop point did not work correctly.

14) If a track was set up to loop from say 30 seconds back to 20 seconds, and while it was looping you turned off looping and changed the stop time to 40 seconds, SMS would get locked up trying to pass the original stop time. This is now fixed, it will continue out to the new stop time.

15) Entering the start and repeat times twice for a track before playing it could result in the track starting from the repeat point rather than the start point. This has been fixed.

****

Version 1.0.67.1
2011-02-23

****

1) There was a day-one bug in crosspoint eq that would have ignored the eq setting, and possibly caused distorted output from the crosspoint. This has been fixed.

2) The memory required for a large crosspoint matrix has been considerably reduced, if the crosspoints do not use eq, as is often the case.

3) Under most conditions processing with a lot of crosspoints is faster than it has previously been.

4) The Machine ID now shows up in the About box whether the dongle is present or not.

5) You can now drag-select the text in the Machine Serial No window in the main window.

6) Previously closing the interface or exiting SMS when an interface with a lot of channels was open could take many seconds. This has been fixed.

7) Eliminated the CONFIG SET SIXFOREIGHT hack that had been added to only use the first 6 channels out of each 8 channels. It turned out the one place that wanted this didn't actually have a use for it, Since it mucked up the innards of things horribly, it went away.

8) Added RESUME as a synonym for REPEAT in the playback TRACK syntax.

****

Version 1.0.66
2011-02-10

****

1) A change put into the audio processing for a special version of SMS caused problems for some users of the standard version. The change has been removed from the standard versions and is only in the special version.

****

Version 1.0.65

****

1) The machine serial number is now logged on startup.

2) A preset load or save command did not allow the word GROUPS, only G or GROUP.

3) When saving or loading group information in a preset you had to specify the type of information you wanted to load or save. Currently the only thing that you can specify is ALL information for the group. The parsing has been changed to default to ALL if the following term is not recognized.

4) Instead of having to specify a range of group channels in a preset command you can now use the word ALLGROUPS to handle all group channels at once. For instance:

    PRESET SAVE GROUP ALLGROUPS ALL TO "GROUP_SETUP";

5) Several bugs in preset command parsing would make perfectly valid commands fail with syntax errors. One case of this was listing groups after channels as in the following example:

    PRESET SAVE CONNECTION MATRIX ALL GROUP G0 - G100 TO MYPRESET

****

Version 1.0.64
2011-01-30

****

1) Updated copyrights to 2011.

2) Added PRESET functionality. This lets you quickly load and save the state of the matrix. You can save a preset with particular gain or EQ settings for a group of channels and then instantly restore them at a later point.

Virtually all parameters for all inputs, playbacks, outputs, or crosspoints can be saved and restored.

Presets are instant. They save the current channel state, or, if a parameter is fading, they save the target value at the end of the fade. When you restore a preset, it instantly sets the various channel values, but it does NOT restore any fades that were in process! Instead, the value is set to what it would have been at the end of the fade.

For playback channels the current stopped/playing state and current track position is NOT saved. If a playback channel has the track info restored the channel will be stopped, and will begin playing at the start point when the channel is resumed. Presets are intended for between-scenes setup, not for running fade changes.

In addition to the matrix state, the ASIO connection info can be saved and restored. If a preset is saved with the ASIO connection info in it, and that info differs from the current matrix state, the currently in use ASIO device will be closed, the new one specified in the preset opened, and the entire matrix will be rebuilt. This will stop all playback and reset all matrix parameters; however, they will be restored to what was saved in the preset after the matrix is rebuilt.

Presets are normally stored in "My Documents\SoundMan Presets". Every preset is stored as an individual file, and the preset files can be given any name the user desires, as long as it is a valid file name.

One preset is recognized with a "special" name. The preset SOUNDMAN_STARTUP is used to load the state of SMS when it is first started. This preset will only exist if the user creates it by saving a preset with that name. The preset must include the ASIO connection info to be useful, as it is loaded at a time when the matrix has not yet been connected. It can also contain setup information for the matrix if non-default setup information is desired. If only the ASIO info is restored, the preset will effectively just open the ASIO device and leave SMS idle, waiting for commands.

Presets can be stored in places other than the normal "SoundMan Presets" directory. This is not recommended, but there are times that it could be useful. There is a command to set a new preset directory. This location will be remembered for the next invocation of SMS, so it only has to be set once, if used. There is a command to display the current preset path.

There are currently four preset commands:

PRESET SET PATH "path"

PRESET GET PATH

PRESET SAVE
[MUTED]
[CONNECTION]
[GROUP group-list ALL]... // can be repeated
[MATRIX | CHAN channel list]
[ALL | MUTE | SOLO | EQ | GAIN | DELAY | chan params]
TO "filename" | SOUNDMAN_STARTUP

PRESET RESTORE
[MUTED]
[CONNECTION]
[GROUP group-list ALL]... // can be repeated
[MATRIX | CHAN channel list]
[ALL | MUTE | SOLO | EQ | GAIN | DELAY | chan params]
FROM "filename"

None of these commands are valid until the ASIO device is opened. That means you cannot save a preset with a close ASIO device state. You also cannot load an arbitrary preset before the ASIO device is opened; only the SOUNDMAN_STARTUP preset will be loaded before the ASIO device is opened.

The commands are described below.

3) PRESET SET PATH "path"

This command will set a non-default preset storage path. This path is remembered in the registry and the same path will be used by all SMS invocations until it is changed.

It is not recommended that the preset path be changed.

4) PRESET GET PATH

This command will display the current preset path.
The format of the one-line response is:

PresetPath="path name"

5) PRESET SAVE

This command saves selected parts of the current SMS state into a preset. The preset can later be restored.

The text following "SAVE" can be divided into three parts:

1. The initial parts MUTED and CONNECTION
2 The repeating list of channels to save data for
2a The types of data to save for all listed channels
3 The preset filename to save the information to.

If the word MUTED appears in the SAVE command, it will be remembered until

the preset is reloaded. During the reload process all of the output channels will be muted, and the mute will only be removed after the preset has been completely loaded. This is intended to prevent loud and embarrassing noises blasting out of speakers if major EQ or delay changes or full-matrix gain changes are being made.

The word CONNECTION results in the ASIO interface data being saved to the preset. This generally should NOT be used except for an initial setup preset of some sort. If a preset contains connection data and you ask that connection data be restored on the reload, the entire matrix is rebuilt, which can take a number of seconds, during which you will have silence and no control over channel parameters (as the channels will not yet exist).

MUTED and CONNECTION can appear in any order after SAVE.

The next part of the command contains an identifier for some channel or group of channels, and then a list of things to save for that group. This can be repeated any number of times, so that it is possible to save different things for different channels.

GROUP will save information for one or more group channels. Currently the only valid information type for group channels is ALL. For instance you could say

PRESET SAVE GROUP G0-G15 ALL

You can give either a single group channel number or a range of group channel numbers (separate by TO or a dash) after the keyword GROUP. If you need to save several disjoint groups, you could for instance say:

PRESET SAVE GROUP G0 ALL GROUP G12 ALL

The word MATRIX, CHAN, or CHANNEL indicates a range of chanels of interest.

MATRIX indicates the entire matrix: all input, playback, crosspoint, and output channels are included in this group. This includes "special" channels such as the signal generators and timecode readers and generators.

The word CHAN or CHANNEL must be followed by a list of channel names and channel ranges. This is much like channel parameters to any command, except that a comma cannot be used to separate the individual channel numbers. Valid channel names and ranges are

I <number>	[-|TO [I|IN|INPUT] <number>]
IN <number>	[-|TO [I|IN|INPUT] <number>]
INPUT <number>	[-|TO [I|IN|INPUT] <number>]
O <number>	[-|TO [O|OUT|OUTPUT] <number>]
OUT <number>	[-|TO [O|OUT|OUTPUT] <number>]
OUTPUT <number>	[-|TO [O|OUT|OUTPUT] <number>]
P <number>	[-|TO [P|PB|PLAYBACK] <number>]
PB <number>	[-|TO [P|PB|PLAYBACK] <number>]
PLAYBACK <number>	[-|TO [P|PB|PLAYBACK] <number>]
X <number>.<number>	[-|TO [X|XPOINT] <number>.<number>]
XPOINT<number>.<number>	[-|TO [X|XPOINT] <number>.<number>]
CROSSPOINT<number>.<number>	[-|TO [X|XPOINT] <number>.<number>]
PX<number>.<number>	[-|TO [PX|PBXPOINT] <number>.<number>]

PBXPOINT<number>.<number> [-|TO [PX|PBXPOINT] <number>.<number>]
PLAYBACKCROSSPOINT<number>.<number> [-|TO [PX|PBXPOINT] <number>.<number>]

In the above lines, <number> indicates a valid number, like 0 or 1001. A quantity in square brackets [and] is optional. So in the first line you can have I0 or I 0 or I0 - I1 or I0 - 1 or I 0 to INPUT 12.

Once you have given a range of channels you have to say what data you want to save for those channels. The same kind of data will be saved for all of the listed channels. Currently you can save the following data types:

ALL	All of the following
CHANINFO	Name, VU flags, MIDI info, TC lock mode
GAININFO	All gain fields, MUTE, SOLO, PHASEREVERSE
DELAYINFO	Delay amount and delay enable
EQINFO	All parametric EQ info for all bands
GEQINFO	All graphic EQ info
PITCHINFO	All pitch shift info
TRACKINFO	All playback track info: name/start/stop/resume/etc.

The simplest way to save all of the information for the entire matrix is

PRESET SAVE MATRIX ALL

But you probably also want to save the group info if you are using groups, so you might say

PRESET SAVE GROUP G0-15 MATRIX ALL

Since this might be noisy as it is restoring EQ and delay info, it would be better to say

PRESET SAVE MUTED GROUP G0-15 MATRIX ALL

If you are going to make an initial setup preset, you would most likely say

PRESET SAVE CONNECTION GROUP G0-15 MATRIX ALL

You can save different things for different channels. For instance, you could say

PRESET SAVE CHAN I0-15 CHAN O0-15 GAININFO MATRIX EQINFO

This would save the gain info for only the input and output channels, but would save all of the EQ info for the entire matrix. When this preset is restored the gain for the input and output channels would be restored, but the gain for the playback channels and the crosspoints would not be changed.

Finally, after all of the data to be saved is given, you need to specify the name of the preset file to save to:

TO <file name>

For instance

TO "Preset 12"

So the total command to create an initial preset could be

PRESET SAVE CONNECTION TO "SOUNDMAN_STARTUP"

or

PRESET SAVE CONNECTION G0-100 MATRIX ALL TO "SOUNDMAN_STARTUP"

6) PRESET RESTORE

This command will restore some or all of the information in a preset. You can select which parts of the preset information you want to restore; you do not have to restore all of the information in the preset.

The restore command has almost exactly the same syntax as the save command. You specify the groups and channels you want to restore, and then the type of information to restore to each selected group or channel.

If the requested information is not in the preset it will not be restored. If there is information in the preset that is not requested it will not be restored either. Consider the following example:

```
PRESET SAVE MATRIX ALL TO "TEST"
PRESET RESTORE CHAN I0 GAININFO FROM "TEST"
PRESET RESTORE MATRIX DELAYINFO FROM "SOUNDMAN_STARTUP'
```

First we save all of the information for the matrix to a preset called TEST. This preset has all of the matrix settings, but not the group information, and not any connection information. It also wasn't saved with MUTED set, so the outputs will not be muted unless the restore has MUTED on it. In our example it does not.

The RESTORE command for the TEST preset will only restore the gain for channel I0, even though the preset has the gain information for all of the channels in the matrix.

The second RESTORE command for the SOUNDMAN_STARTUP preset will restore all of the delay values for all channels in the matrix. The ASIO connection info will not be restored, even though it is present in the preset. Neither will the group information, or any other channel property except the delay values and delay enables. Because the preset specified MUTED when it was created, the outputs will be muted a moment while the delay values are being restored. If it had not, then MUTED could be given on the RESTORE command to force the outputs to be muted during the restore.

7) Re-ordered the main window startup code to make is look better while it is searching for the dongle.

8) If the main window started minimized the toolbar icon menu still defaulted to "hide window" even though the window was already hidden.

9) A new command has been added to set or clear the SINGLE THREADING debug flag in the main window. When set, this forces all audio processing to be done on the ASIO callback thread directly. With some bad ASIO drivers this can result in removing clicks and pops. However, it limits the amount

of audio processing that can be done to the amount that can be accomplished on a single processor.

The command

CONFIG SET SINGLETHREADING [ON | OFF]

This can be interrogated with

CONFIG GET SINGLETHREADING

which will return a response of

"SingleThreading = ON;" (or OFF)

This should only be done if absolutely required. It is not required for most ASIO drivers. Since it limits processing power, it can also result in distorted audio if you exceed the amount of work that can be done on a single processor.

10) Fixed problems with the splash screen not appearing as a result of changes to make the tray menu work correctly.

11) Improved the layout of the Ecoustic splash screen and used a different logo bitmap that looks somewhat better.

12) Improved the processor scheduling routine out of the ASIO callback to get better use of a multiprocessor system.

13) The "Ecoustic" node in the folder path was incorrectly being created in the root of the C drive.

14) There is now callback routine processor usage information in the main window, and it is also logged to the log about every 100 seconds.

15) On some machines with poor inter-processor interrupt design, SMS will tend to crackle and pop a lot. Restricting the processing to a single core can sometimes help on this sort of system. There is a new command that allows you to single-thread the ASIO audio processing.

CONFIG SET SINGLETHREADING [ON | OFF]

This will set ASIO Callback single-threading on or off. The default (and only value) for a single processor system is ON. On multi-processor systems, the default is OFF, allowing the processing to be distributed over all of the available processors. Turning single-threading on will restrict the processing to a single processor, which can help if the system has a problem handling IPC interrupts quickly.

This command is only valid after the ASIO interface has been opened.

You can also interrogate the status of the single-threading option:

CONFIG GET SINGLETHREADING

This command will return the current state of the single-threading flag.

The response looks like

SingleThreading = ON; or SingleThreading = OFF;

This command is only valid after the ASIO interface has been opened.

16) Added a button to the interface to reload the initial setup preset without having to stop and restart SMS.

****

Version 1.0.63

****

1) Fixed a day-one bug that under rare timing conditions could cause SMS to crash. In the process, managed to speed up some of the processing by a fair amount.

2) Added an exception handler to the GenerateData function in the input converters to prevent a crash that could occur in very rare circumstances.

3) The SET CHAN IN xx PORT <name> command was not being parsed correctly.

4) Fixed a timing window that could cause a crash in CommandCue mode.

5) Setting a timecode generator using a numeric channel number was broken. This is now fixed.

6) Added the machine serial number and the dongle id (if the dongle is present) to the main dialog where most anyone should be able to find them.

7) Fixed a problem where the dongle serial number could display with incorrect leading FFFFs.

8) SMS would try for 60 seconds to create the shell notify hook so that it could do balloon popups from the system tray. If a system is running without a shell (probably for security reasons) then this creation will never succeed, and 60 seconds will have been wasted. Cut the time down to 4 seconds. This will hopefully give enough time on an auto-logon at boot time for the shell to get it's mind in order, while not taking a huge amount of time.

9) Now log more license info in the log to help in problem debugging.

10) If a file was locked to timecode and the timecode ran long enough that the file played off the end, the file would be marked stopped. If the timecode jumped back into the file range, the file would not restart playing until a new PLAY command had been issued to the channel or group. This is now fixed, If a channel's position is locked to timecode and the timecode re-enters the file range, the file will start playing again, even if it had previously stopped.

11) The parameters for an 8192-long FFT were not set up correctly and would result in a crash if you tried to use one that size. This affected pitch shifting and the graphic equalizer functions.

12) An OEM Ecoustic version has different branding and contact info. All dumps and logs will go into an "Ecoustic" directory in My Documents.

13) Updated the copyright in the About box to include 2010!

****

Version 1.0.62
2010-07-08

****

1) The "Syncstart" information message will now show a repeat time of -1 instead of 0 when there is no repeat point set. This makes it easier to tell when looking at a log if the track will loop.

2) When a track loops, there is a log message to show that the track looped. This also makes analyzing logs easier. Note that the log messages are queued by the playback code, since they can't afford the time to make a log message at the time the actual loop occurs. So the 'looped' log messages may have a timestamp that is a number of milliseconds after the actual loop happened. In particular, when a group of tracks is playing in unison, the individual tracks will show 'looped' messages, and they will probably all have slightly different timestamps. This is NOT an indication that the tracks are out of sync.

3) In the last version, a group of tracks might loop even though it was only supposed to play once. This is fixed.

4) There were a couple of long-standing bugs that could cause crosspoints to have random amounts of delay in relation to each other. For a long time this wasn't noticeable because the crosspoint delays were disabled by default. However, several releases back they were changed to be enabled by default since ABShowMaker has no way to control crosspoint delay enable. Especially in installations with large ASIO buffer sizes, this made the occasional random delay quite noticeable.

5) Setting a delay on a timecode generator did not work properly. The signal would be delayed after the generator started, but the signal stopped instantly when the generator stopped, rather than continuing for the delay period as it should have.

6) Several new GET requests have been added to get information about a loaded or playing track. All of these get static information about the file that is loaded onto the channel. If no file is loaded, the attributes that return strings will return empty strings, and the attributes that return numbers will return zero.

//	TITLE	# TrackTitle	chanid="title"
//	DISPNAME	# TrackDispName	chanid="name"
//	SUBJECT	# TrackSubject	chanid="name"
//	COPYRIGHT	# TrackCopyright	chanid="name"
//	COMMENT	# TrackComment	chanid="name"
//	SAMPLERATE	# TrackSR	chanid=fpt
//	TRACKS	# TrackTracks	chanid=number

GET CHAN channel TRACK TITLE will get the title information from the file if there is any title data present. Otherwise it will return the file name without the path or "wav" suffix.

GET CHAN channel TRACK DISPNAME will get the display name attribute from

the file, if it exists.

GET CHAN channel TRACK SUBJECT will get the subject attribute from the file, if it exists.

GET CHAN channel TRACK COPYRIGHT will get the copyright attribute from the file, if it exists.

GET CHAN channel TRACK COMMENT will get the comment attribute from the file, if it exists.

GET CHAN channel TRACK SAMPLERATE will get the sample rate for the file.

GET CHAN channel TRACK TRACKS will get the number of tracks or channels in the file.

For all of the GET commands above, the response will be in the form of a line starting with the keyword shown above (for instance TrackTitle) followed by one or more instances of <channel id>=<value>. A typical GET and the response might be:

```
GET CHAN P0-P1 TRACK TRACKS
TrackTracks P0=1 P1=4
```

7) The GET GENERATOR and SET GENERATOR commands to get and set signal generator parameters will now accept a valid signal generator channel number like IN 1000. Previously only the channel number was valid. This should eliminate some amount of confusion in using these commands. The bare number format is still valid.

8) GET TCGEN START TC now returns the starting time value and offset as timecode values rather than floating-point numbers as was erroneously done before. The same values can still be obtained as floating-point numbers by using GET TCGEN START TIME. Previously both commands returned the data in the same format.

```
//      START  TIME                  # StartTime=fpt At fpt
//      START  TC                    # StartTC=tc At tc
```

The above lines show the response format for the GET TCGEN START TIME and GET TCGEN START TC commands.

9) The number of SMPTE timecode readers, SMPTE timecode generators, and signal generators are now controlled by the license info in the dongle. Previously there were always two of each created, regardless of the info in the dongle. Current license keys will default to two of each so that current users will not see a change from this.

10) The syntax of the GET VU command has been changed slightly to allow monitoring either pre or post VU on any control node.
The new syntax is:

```
GET VU [PRE|POST] <channel list>
```

If neither PRE nor POST is given (the old syntax form is used) then POST VU readings will be returned as is normal. If POST is given the same post VU

readings will be returned. If PRE is given then PRE VU readings will be taken before the control point has an effect on the signal. This might be most useful for input channels.

When switching from pre to post VU or vice versa, the accumulating VU averages are cleared, so the first reading after changing will return zero for the average and peak values for all channels. It is advisable to NOT switch between pre and post VU readings frequently in an attempt to get both values from a single channel. It will not work well. It is best to only switch on a user request.

11) A new SET VU command has been added. Getting VU values uses quite a lot of processor time. If you got VU values for the inputs at one time but now no longer need these values, it would be best to turn off the VU accumulation code on those inputs. This is even more important when getting VU readings on crosspoints since there are so many of them.

SET VU [PRE|POST] channel-spec ON|OFF

You can specify whether you want to turn on or off pre or post VU processing for the range of channels. Only one type can be on at any time, so if you had pre VU on and now turn on post VU processing, the pre VU processing will be turned off automatically.

If you do not specify PRE or POST and use ON, the command will return an error, since only one type can be on at once. If you do not specify PRE or POST and give OFF, all VU processing will be turned off for the range of channels.

12) A "GET GROUP" command has been implemented to get information about groups.

//	GROUP group-spec	#	
//	NAME	# Name	chanid=none|name
//	CHANS|CHANNELS	# Channels	chanid=chanid|none
//	TCREADER	# TcReader	chanid=chanid|none

GET GROUP channel-list NAME will get the name of the group.
This is the same as GET CHAN channel-list NAME.

GET GROUP channel-list CHANS or CHANNELS will get a list of the channels assigned to the group. Note that the response is the group channel id, an equal sign, and then a space separated list of the individual channel ids. The next group, if more than one was requested, will have the group id and an equal sign before its list of channels. If a group has no channels assigned, the first and only channel name will be "none".

GET GROUP channel-list TCREADER will get the channel ids for the timecode readers assigned to the channels in the group. If a group does no have a timecode reader assigned to it (as is often the case) the reported channel id is "none" (without the quotes). The response has the channel id of each requested group and the matching channel id (or "none") for the reader, following the equal sign.

13) The SET GROUP group-id TC READER <reader-id> command will now allow either a numeric channel number for the timecode reader as before, or a full channel specification for the reader such as OUT 1000 or a channel name.

14) The routines that looked up channel names would fail to find named "special" channels like signal generators and timecode readers.

15) Added a GET FILE command to let you interrogate attributes of an audio file such as the number of tracks and the sample rate, plus some of the internal data items.

```
//      GET FILE "filename"         # FileInfo "name"
//      SAMPLERATE                  #       SampleRate=number
//      LENGTH                      #       Length=number - time in seconds
//      TRACKS                      #       Tracks=number
//      FORMAT                      #       Format=WAVE|AIFF|RAW
//      TITLE                       #       Title="string"
//      COPYRIGHT                   #       Copyright="string"
//      COMMENT                     #       Comment="string"
//      SUBJECT                     #       Subject="string"
//      DISPLAYNAME                 #       DisplayName="string"
```

The filename to be interrogated follows GET FILE, and should be quoted. Following the file name are one or more attribute names to be interrogated. If the file can be found and opened, the response will begin with FileInfo followed by the quoted name of the file, and then the attribute key words followed by equal signs and the attribute values.

16) Added a GET TCREADER command to let you get information about a timecode reader.

```
//      GET TCREADER channel        # TcReader chan
//              CODE|SHAPE          #       Shape=Sine
//                                  #               SMPTE_30
//                                  #               SMPTE_30_Drop
//                                  #               SMPTE_25
//                                  #               SMPTE_24
//                                  #               IRIG_A_Pulse
//                                  #               IRIG_A_Pulse_Inverted
//                                  #               IRIG_A_Mod
//                                  #               IRIG_A_Mod_Inverted
//                                  #               IRIG_B_Pulse
//                                  #               IRIG_B_Pulse_Inverted
//                                  #               IRIG_B_Mod
//                                  #               IRIG_B_Mod_Inverted
//                                  #               Unknown
//              POSITION            #       Position=fpt    in samples
//              TIME                #       Time=fpt        in seconds
//              TC                  #       TC=timecode     in timecode
//              FREEWHEEL           #       FreewheelCount=num
                                    #       same as GET FREEWHEEL
//              DATE                #       Date=yy/mm/dd
//              DAY                 #       Day=number
//              USERBITS            #       Userbits=8_hex
//              SPEED               #       Speed=fpt
//              RUNNING|PLAYING     #       Running=YES|NO
```

Only one timecode reader can be interrogated with a single command. You can get the current shape of the timecode being read, if it is

recognized. If timecode has stopped, the code shown will be the last code recognized.

POSITION, TIME, and TC all return the same information in different formats, that is, the current timecode value being read (or the last timecode value read if timecode has stopped).

FREEWHEEL returns the current freewheel count.

DATE will return the same thing as USERBITS for a SMPTE decoder. It will return the date (usually a Julian date) in the IRIG code for an IRIG decoder.

DAY will return 0 for a SMPTE decoder and the Julian day value from the code for an IRIG decoder.

17) The ASIO driver input and output buffers are now being cleared when the ASIO device is opened. The drivers are supposed to clear their own output buffers, but not all drivers were doing this. This resulted in nasty noise for an instant when first starting.

18) Added code to keep the UI from hanging completely while polling for the dongle at startup. This should keep SMM from wanting to take a dump because it believes (correctly) that the interface is hung.

19) Found and fixed several small problems with delay fades that occurred at intermediate rates of speed. There were separate problems on the output channels and the crosspoints.

20) Looking for the dongle on startup takes 20 seconds if there is no dongle (demo mode startup). The interface used to be hung during this time, resulting in nasty messages in the log in SoundMan-Monitor, and possibly even requests from SMM to take a program dump to analyze the hang. The interface will no longer be hung while looking for the dongle, so SMM will not log nasty messages and try to take dumps.

****

Version 1.0.61
2010-06-21

****

1) It is now possible to loop time on a timecode generator. A timecode generator looks like a playback channel. So you set the looping parameters just as you would for a track. For instance, to loop the first timecode generator from 10 seconds back to 0 seconds, you can enter:

    SET CHAN P1000 TRACK STOP TIME 10 REPEAT TIME 0;

    You can also set the initial start time for the timecode generator with the TRACK START TIME parameter.

2) You can now enter a normal timecode generator channel specification like P1000 for the SET TCGEN command. Previously you had to enter the bare number "1000" to do this. You can still enter a bare number, or no specifier at all, which will default to the first timecode generator. This was fixed in the previous release for GET TCGEN.

3) Changed the "timecode reader freewheel timeout" message to be an Information message rather than a Warning message.

4) Added a command to hide (minimize) or show the main UI window. On startup the main interface and the splash screen can be suppressed by adding the /NOSPLASH parameter to the command line starting SoundMan-Server. However, if for some reason that is inconvenient, the window can be made to hide in the tray by sending CONFIG SET WINDOW MINIMIZED. The window can be made visible again by sending CONFIG SET WINDOW VISIBLE.

5) Now log when the user requests display of the ASIO driver control panel.

6) No longer stop ASIO processing when getting an ASIO reset request from the driver. This should in most cases eliminate problems with audio disappearing after looking at the control panel for a driver and adjusting it, at least with decently written drivers. If a driver attempts to reconfigure itself on the fly before we do a complete reset (in violation of the ASIO spec) then there is a slightly better chance that we will immediately crash.

    Since the driver may have internally stopped processing audio, it is still necessary to close and re-open the driver as soon as possible to allow the new driver configuration to take effect.

7) Now show a nasty tray popup balloon for 30 seconds when we get an ASIO reset request, describing what needs to be done.

8) Made a small change that may cure the problem of having random delays on playback at startup.

9) GET CHAN list TRACK POSITION will now return the position values as an integral number of samples. Previously this would return a floating-point number with less accuracy in most cases.

10) When SMA is started minimized the tray menu default will now correctly be the show the window, not to hide it.

11) Fixed a crash that could occur when closing SMS while audio was playing.

12) A Syncstop message will no longer be logged is a STOP command is issued to a channel that is already stopped.

13) Clearing a resume point on a channel that doesn't have a track loaded will no longer create a warning message in the log.

14) If you muted the crosspoints the output VU would hang at the current value rather than properly decaying to zero. This is now fixed.

15) There is a new LOG command to let the user add comments to the text output from SMS. This can be useful for documenting problems or other reasons. The log records are timestamped and included in sequence in the history.

```
//      LOG
//              USER                    "text";         // U: text
//              INFO|INFORMATION        "text";         // I: text
//              WARN|WARNING            "text";         // W: text
```

The LOG command is followed by a keyword that indicates the kind of log entry to make. All log entries begin with a date and time stamp, followed by a single letter and a colon that identifies the type of the log record. The character will be one of the following:

C	Command	The text of a command to SMS
R	Response	The text of the response to a command
D	Debug	A debug output message, not common in release software
I	Information	A line of information
U	User	A LOG command marked "USER"
W	Warning	A line indicating an unusual or error condition

After the word indicating the type of log entry to make is the text of the log entry. This can be a single string in quotes, or it can be a series of individual words, ending in a semicolon or the end of the input line.

Log entries in no way change how SMS works or what it is doing. All they do is allow the user to make their own entries in the captured log data.

****

Version 1.0.60
2010-05-25

****

1) The GET TCGENERATOR command with no parameters other than the timecode generator number returned a useless response. It will now be equivalent to GET TCGEN <number> TIME and return the current generated time from the generator. Note the time is returned as a floating point number of seconds and not as a timecode. To convert the fractional seconds to frames you can get the code parameter from the timecode generator, which will tell you the frames per second, and a multiply by the number of frames per second will convert the time to frames.

2) If a wave file had an incorrect format and showed the whole file length as the RIFF chunk length it would not be parsed. This formatting error is now handled quietly and the file will load correctly if there are no other formatting errors.

3) If wave file header parsing failed, the file load would succeed, but the file would show a length of 0. The load will now correctly fail with a useful error message, making it more obvious that something is wrong.

4) The GET TCGEN command will now accept a channel number in the normal format of "P1000" rather than requiring a bare number of "1000".

5) A parsing error in GET TCGEN that caused many parameters to be ignored has been fixed.

6) Garbage syntax on the end of a GET command was ignored rather than returning an error as it should have done.

7) The timecode generators are now initialized to generate SMPTE at 30fps by default. Previously they defaulted to generating 1KHz sine wave tones, which was confusing.

****

Version 1.0.59
2010-05-17

****

1) Track start, stop, and repeat times could be entered that were negative. For the stop and repeat times this would have disabled the stop or repeat point, but for start times, this would often result in a crash when starting to play. Negative times are now properly detected as invalid and an error message will result.

2) Startup was hanging with the AudioScience Cobranet driver. Changed the order of some code to work around it.

3) In some cases with many channels we could fail to get memory we needed but not notice the failure. Now if we can't allocate all of the channels we will return an error.

4) Added a SIXFOREIGHT option to CONFIG SET INTERFACE to skip the last two channels out of each group of 8 channels. This is used by SMA in an unusual Cobranet configuration where each bundle only has six channels.

****

Version 1.0.58
2010-04-24

****

1) A code bug caused the timecode reader to freewheel forever if timecode was lost. This is now fixed.

2) Warning and information log messages have been added to show when the TC reader loses timecode, sees a constant repeated (stopped) timecode value, and when timecode again begins moving. All of these messages include the timecode value and reader number.

3) A new command has been added to set the freewheel count for a timecode reader. The freewheel count defaults to 5 frames, and can be set anywhere from 0 to 500 frames, or to INFINITE.

   The new command syntax is:

   SET CHAN chan FREEWHEEL n | INFINITE

4) A new command has been added to get the freewheel count value for a timecode reader:

   GET CHAN chan FREEWHEEL

   The returned value is either

   Freewheel On=count

   or

   Freewheel On=INFINITE

   Where "On" is "O" for OUTPUT followed by the reader channel number, and the count is the number of freewheel frames, 0 to 500.

5) The SMPTE timecode reader was accidentally leaving a debug log file on the C drive. This has been removed.

****

Version 1.0.57
2010-04-14

****

1) If the tray notification icon did not create correctly we would exit the program. This was unnecessary and has been removed.

2) It seems that creating the icon could fail on Windows Embedded if SMS was started immediately on system power up. SMS will now retry for up to 20 seconds to create the tray icon. This seems to get around the timeout response being returned by Windows.

3) SMS was not correctly parsing wave file headers that contained invalid or unexpected items in a LIST chunk. This resulted in the file being seen as zero size, so it wouldn't play.

****

Version 1.0.56
2010-04-05

****

1) Updated the dongle code to match changes in SMA.

2) Now check for up to 20 seconds for the dongle being present when starting. Previously if SMS was auto-started from the startup folder on boot-up and auto-logon, there was a timing window that could occasionally result in not finding the dongle even though it was present.

3) A really annoying VU meter interaction with SMA that caused the VU meters to flash has been fixed.

4) Fixed the copyright date to 2010 in the splash screen.

5) There could be a delay on some of the playback channels, even though checking the delay time for the channel would return zero. Setting the delay to zero would clear this phantom delay. This problem should now be fixed, and no phantom delays should be possible.

6) Found and fixed a small bug in the code that sets a cross-mute on other input channels when one input channel is soloed.

****

Version 1.0.55
2010-03-18

****

1) There was a timing window in the dongle code that could result in a deadlock on a multiprocessor machine.

2) Updated the dongle code to match changes in SMA.

****

Version 1.0.54
2010-03-02

****

1) Disabled the "TCP received XXX bytes" debug message that was putting unneeded noise into the log file.

2) There is no longer a "beep" sound when the tray icons for various status messages and warnings are displayed. This can prevent bad sounds coming out of speakers if Windows is sharing the show audio hardware.

3) Fixed a timing window that could cause crashes on some machines.

4) Track start/stop/repeat/loop points were incorrectly being set using the interface sample rate rather than the file sample rate. When playing 44100 wave files using a 48K interface this could result in the times being considerably off from what was actually requested.

5) Setting a repeat point that was before the start point on the track would not loop correctly back to the desired point, the track just stopped at the end. This is fixed.

6) The fix for early starts broke normal late starts. That is now fixed.

7) If you tried to resume a track that had finished playing you would get hung waiting for a track to go ready that was never going to go ready.

****

Version 1.0.53
2010-02-04

****

1) Updated the copyright to 2010.

2) Changed things so that all 64 output channels are now available from SoundMan-Assistant.

3) Fixed a problem in the timecode generator that was preventing it from generating an offset time value when locked to an incoming timecode stream.

When setting up the command sequence to lock an offset generator to timecode the order of the commands is important. The following sequence will generate a timecode offset by 1 hour from the incoming timecode:

```
set tcgen 1001 code smpte30
set group g0 chans p1001 tc reader 1000 lock time 0;
set tcgen 1001 start tc 1:0:0:0 at time 0;
```

The first command sets the code of the timecode to be generated. This needs to be done before the generator needs to generate anything, so is best done before the generator channel is added to the timecode reader group.

The second command adds the generator (channel P1001) to a timecode reader group, selects reader 1000 (channel OUT 1000), and sets up to lock to a timecode chase beginning at time 0 (that is, no hour offset).

The third command MUST be after the SET GROUP command or it will be ignored. It tells the timecode generator to generate time starting at 1 hour when it has an incoming timecode of zero hours. Thus the generated time will be one hour ahead of the time read.

If ONLY the timecode generator is locked to the timecode reader (just regenerating timecode) then this set of commands can be simplified:

```
set tcgen 1001 code smpte30
set group g0 chans p1001 tc reader 1000 lock tc 1:0:0:0 at time 0;
```

This will have the same effect as the previous set of commands IF only the tc generator is locked to the reader. However, if audio files are also locked to the reader, this will have the undesirable effect of starting the audio files 1 hour into playback, which will probably be past the end of the files.

To keep from having audio offset problems the initial command sequence is strongly recommended when offset timecode is needed.

****

Version 1.0.52
2009-11-11

****

1) Externally requested dumps were not being taken. They now are.

2) A popup message after an externally requested dump or a UI timeout would cause SMS to hang until the user OKed the popup. This was a bad idea, and the popup will now only occur if a serious fault occurs in SMS, not from a timeout or other recoverable problem.

3) The "SoundMan Dumps" directory will be compressed if it isn't already.

4) Updated copyright on splash screen to 2009, which should have been done about a year ago.

5) Modified the TCP receive logic to hopefully be more robust in the face of possible errors. Added logging of error conditions that would otherwise have gone unreported.

6) The debug interface is now the same width as the SMA debug interface. This makes things look better if both windows are visible.

7) Fixed a problem in the code that checked at startup if the dump file directory was compressed.

8) Added code to check periodically if there is any data waiting on the TCP port and process the data. This may help if we are getting lost interrupts on input processing.

9) Corrected a misformatted TCP error message.

10) Added code to try to keep from blocking the user interface when waiting to send a response to the TCP command requestor.

****

Version 1.0.51
2009-10-18

****

1) Fixed a severe meter layout problem when starting in iconic mode. Meters laid out correctly if SMS was started with the window displayed.

****

Version 1.0.50
2009-10-13

****

1) Removed some old unused dialog templates to simplify code.

2) Fixed a problem with VU meter layout that could cause a crash on exit.

3) Control input and output may now be entered by either TCP port 20000 or UDP port 20000. When using UDP multiple commands separated by a carriage return and linefeed ("\r\n", or in hex 0D0A) can be given in a single packet. The first command in the packet starts at the front of the packet. A command can not be split across packets. The maximum packet size is 1500 bytes.

4) The command line entry window in the SMS window no longer requires a semicolon to recognize the input. A semi-colon still ends an input line but it is not necessary. A carriage return will also work. Previously carriage returns were ignored.

****

Version 1.0.49
2009-10-01

****

1) The routines to calculate the machine serial number were not the same in ShowMan, SMA, SMS, and the E-Show tools. This release has consistent routines that will produce the same serial number.

****

Version 1.0.48
2009-09-23

****

1) If there were more than 32 output channels, every time you opened the ASIO device the window would get taller by one or more rows of VU meters. Now the window is correctly sized to the VU meters.

2) The SMS window now originally comes up without the blank space for a row of VU meters. The correct space will be added when an interface is opened. When the interface is closed all of the extra space will go away again.

3) Added debugging code to check for smashed memory around the areas of opening and closing the ASIO drivers. Several ASIO drivers have been found that seem to have memory corruption problems.

4) Some drivers could produce errors that would result in the watchdog thread killing itself when it was trying to kill an overtime callback thread from the driver. This would hang SMS. This has been fixed.

5) Fixed a usually benign memory corruption problem that could occur when using an ASIO device that uses 24 bit sample format. One byte past the end of the output buffer would be corrupted.

6) Now only display the last two digits of the channel number over the VU meters. Previously with channels of 100 and over the number was centered and difficult to read.

7) Previously it was difficult to get more than 64 channels of VU meters to be seen, and almost impossible to get more than 96 visible, even on a very large desktop screen layout. The VU meters will now shrink vertically as needed to insure that they fit in a reasonable amount of screen space.

8) Fixed a memory leak that occurred when exiting the program and using multiple ASIO drivers in an ASIOGROUP statement.

9) Fixed some problems with allocating and deallocating asiogroups. While the old code worked in most cases, some drivers would fail to close correctly and would not reopen after being closed.

10) Fixed several problems with asiogroups possibly getting locked up while processing audio. In extreme cases this could require a PC reboot to clear the device and let it make audio again.

11) The processor and disk usage monitors could blank when they reached a value of 100%, rather than simply showing a full bar. This has been fixed.

12) Holding the shift key down while clicking the X (close) gadget in the upper right of the title bar will cause the program to exit rather than to minimize. Clicking the X without holding the shift key will minimize in the normal way.

13) There was an error in the calculation of the machine serial number that

could keep temp dongle files from working.

****

Version 1.0.47
2009-08-31

****

1) Changed some code in the temp dongle file reading logic to correct some minor errors that were preventing temp dongle files from working correctly.

2) Added code to work around ASIO drivers that provide a bad preferred buffer size value, but do give reasonable minimum and maximum buffer sizes.

3) Additional info on the DROPSPEED timecode reader parameter added in the previous release:

Setting DROPSPEED results in the time values from the timecode reader being multiplied by 1.001. This corrects the speed error, but it also results in shifting the time value from the reader. For instance, when the reader reads 01:00:00:00 (one hour) it will actually report 01:00:03:18 at 30fps. If you have audio tracks that start at specific time offsets, you may need to slightly shift the start time value to compensate for this.

This chart shows the desired time in hours and the shifted time value for various times and framerates.

Desired	24 FPS	25 FPS	30 FPS
00:00:00:00	00:00:00:00	00:00:00:00	00:00:00:00
01:00:00:00	01:00:03:14	01:00:03:15	01:00:03:18
02:00:00:00	02:00:07:05	02:00:07:05	02:00:07:06
03:00:00:00	03:00:10:19	03:00:10:20	03:00:10:24
04:00:00:00	04:00:14:10	04:00:14:10	04:00:14:12
05:00:00:00	05:00:18:00	05:00:18:00	05:00:18:00
06:00:00:00	06:00:21:14	06:00:21:15	06:00:21:18
07:00:00:00	07:00:25:05	07:00:25:05	07:00:25:06
08:00:00:00	08:00:28:19	08:00:28:20	08:00:28:24
09:00:00:00	09:00:32:10	09:00:32:10	09:00:32:12
10:00:00:00	10:00:36:00	10:00:36:00	10:00:36:00
11:00:00:00	11:00:39:14	11:00:39:15	11:00:39:18
12:00:00:00	12:00:43:05	12:00:43:05	12:00:43:06
13:00:00:00	13:00:46:19	13:00:46:20	13:00:46:24
14:00:00:00	14:00:50:10	14:00:50:10	14:00:50:12
15:00:00:00	15:00:54:00	15:00:54:00	15:00:54:00
16:00:00:00	16:00:57:14	16:00:57:15	16:00:57:18
17:00:00:00	17:01:01:05	17:01:01:05	17:01:01:06
18:00:00:00	18:01:04:19	18:01:04:20	18:01:04:24
19:00:00:00	19:01:08:10	19:01:08:10	19:01:08:12
20:00:00:00	20:01:12:00	20:01:12:00	20:01:12:00
21:00:00:00	21:01:15:14	21:01:15:15	21:01:15:18
22:00:00:00	22:01:19:05	22:01:19:05	22:01:19:06
23:00:00:00	23:01:22:19	23:01:22:20	23:01:22:24

****

Version 1.0.46
2009-07-31

****

1) If the number of output channels is a multiple of 16, the VU meters will be laid out in rows of 32 columns. This will make 64 channels fit into two rows, rather than the three rows currently.

   If the number of channels is more than 32 and not a multiple of 16, the VU meters will be laid out in rows of 30 columns, as they were in previous versions. Less than 32 output channels will have the VU meters laid out in a single row, as previously.

2) Previous versions had a bug that prevented output VU meters in rows past the first row from working. This is now fixed.

3) The timecode reader checked the wrong bit in the SMPTE frame to determine the difference between 30fps and dropframe. The result was that dropframe timecode was decoded as "slow 30fps" timecode. This has been fixed.

4) Sometimes people record timecode in strange ways. A classic case is recording plain 30 fps timecode at an actual rate of 29.97 fps.

   The timecode reader can only accurately detect 4 framerates by itself:

   30 fps
   29.97 dropframe
   25 fps
   24 fps

   It is important to note that the difference between 30fps and 29.97 dropframe is that the SMPTE frame has the "drop frame" bit set in the frame. If the dropframe bit is set the timecode is expected to appear at 0.999 times the normal rate, which is the difference between 30fps and 29.97 fps.

   But if someone records 30fps NON-DROPFRAME timecode at 29.97 fps, SMS cannot detect that the nominal framerate is 29.97 rather than 30. This is only 0.1% slow, and a movie projector generating SMPTE can easily be off by +- 5%. So it merely looks like normal code that is a little slow, and as a result the timecode reader generates time that is a little slow. This isn't very good if you are trying to sync audio to that 29.97 fps source, as the audio will play slow and drop behind the image.

To get around this problem, new syntax has been added to SET GROUP to allow you to specify that you know the timecode is slow, and then SMS can compensate for the slow rate.

```
SET
        GROUP group-spec
                NAME <name>
                CHANS | CHANNELS channel-spec
                CLEAR
                TC | TIMECODE
                        READER          {channel number}
                                DROPSPEED
                                NORMALSPEED
                        LOCK
                                SPEED
                                {TIME|TC} <timecode> [AT {TIME|TC] value]
                        UNLOCK
```

The SET GROUP command to set up a number of playback channels on a timecode reader now looks like the above. The DROPSPEED and NORMALSPEED parameters have been added. Note that these MUST come right after READER and the reader channel number!

DROPSPEED tells the timecode reader that the timecode is running at "dropframe rate", or 0.1% slower than the nominal rate. Use this when you have plain 30fps timecode recorded at 29.97.

NORMALSPEED is the default and tells the reader that the timecode is running at the nominal rate. Use this when 30fps timecode is running at a real 30fps, or when you have dropframe-coded timecode running at 29.97 fps. If you do not put either DROPSPEED or NORMALSPEED, then NORMALSPEED is assumed.

An example of normal 24/25/30fps or 29.97 drop timecode setup:

```
SET GROUP G0 CHANS P0-3 TIMECODE READER 1000 LOCK TC 1:0:0:0;
```

An example of locking to 30fps timecode running at 29.97 fps:

```
SET GROUP G0 CHANS P0-3 TIMECODE READER 1000 DROPSPEED LOCK TC 1:0:0:0;
```

****

Version 1.0.45
2009-07-14

****

1) Fixed a playback lockup that could occur under certain conditions when many commands were issued at once.

2) Setting a resume point that was past the end of a track would cause SMS to loop. This is now fixed. A resume point at or past the end of a track will be ignored. A warning message is generated in the log.

3) There is now an info message in the log when a track actually stops as well as when it starts.

****

Version 1.0.44
2009-05-31

****

1) A broken ASIO driver installation with a missing code file could cause SMS to crash on startup. This is now handled. (This was a day-one bug in Steinberg-supplied ASIO driver code!)
A warning message will appear in the log indicating the ASIO name of the bad driver installation.

2) Removed extraneous newline from "Mapped channel ordering" log message.

3) Creating a dump file could fail, so it is now protected in a try block. Hopefully this will catch the failures.

4) Added code to retry a dongle read if it fails, since that seems to happen now and then.

5) Added some protection logic to the audio file read routines as they could sometimes fault under unusual conditions.

6) Crosspoint delay setting was badly broken in a number of ways. This didn't show up with the small buffer sizes we usually use for testing the audio path, but was blatantly obvious on large buffer sizes.

7) Some problems also existed with delay fades on input and output channels when using large buffers.

8) Added pitch shifting ability to output channels. Previously it was only available on input channels, although the commands appeared to work for all types of channels.

****

Version 1.0.43
2009-05-12

****

1) Setting phase reverse on a crosspoint would incorrectly mute the channel. This now works correctly.

2) If you attempted to PLAY a group before defining the channels for the group things would mess up and refuse to let the group be used ever again. This is now fixed.

3) It is now possible to use multiple ASIO interfaces at the same time, with some important restrictions:

   1 All ASIO interfaces MUST have the same number of samples in the buffer size. For example, all must be 64 samples or all 512 samples. One driver at 64 samples and one at 512 samples will not work.

   2 All ASIO interfaces MUST have the same sample rate, or be possible to set them all to the same rate. This means they must all do 48000 or 44100, unless you specify some other specific sample rate, in which case all of the drivers must run at that rate.

   3 All of the physical interfaces MUST be sample-locked using Wordclock or something similar. The interfaces will run if they are not sample locked. But as with all digital audio, if the source and destination do not have the same clock you will get clicks and pops. In the case of SMS you can also get phase cancellation if you play the same source through multiple interfaces and they are not sample-locked.

   4 The interfaces do not have to use the same sample data format. It is possible to mix drivers that use 24 bit samples and 32 bit samples, or even 16 bit samples. If you mix ASIO devices with 16 bit samples with devices using 32 bit samples the difference in audio quality will be immediately obvious, so this is not recommended.

   5 Grouping interfaces adds one buffer time of delay to the output path. Therefore latency will be longer by the time it takes to process one buffer of data. You can compute this time by:

      added delay = buffersize / samplerate

      For instance, a buffer size of 64 samples will add 1.3ms, which is probably not enough to worry about. But a buffer size of 2048 samples will add 43ms, which is a significant delay and would make it impossible to use the outputs to drive stage monitors. If the output audio is synchronized to video this is nearly 4 frames times at 60 frames/second, and would be very obvious. You would have to adjust the timecode following offset to compensate for this.

   6 If several ASIO devices are grouped for use together, they CANNOT be used separately while the group exists, even if the group is not currently in use. you must destroy the grouping before the devices can be used individually. Of course, the group can be recreated later if

needed.

7 A group CANNOT be created if one of the interfaces to be used in the group is currently in use. You con only create a group from interfaces that are not currently in use.

8 A group of interfaces has a name. This name CANNOT be the same as the name of an existing interface or interface group.

9 You can only create a group from actual ASIO interfaces. You CANNOT create a group by combining another group.

To group multiple ASIO devices you use the command

CONFIG SET ASIOGROUP "name" INTERFACE {"name"|number}...

For instance with two existing interfaces they will be numbered 0 and 1.
You could create a group with:

CONFIG SET ASIOGROUP "MYGROUP" INTERFACE 0 INTERFACE 1

This will group both interfaces together as a single interface group,
letting you use all of the channels on both interfaces at the same time
(if your license limits will allow that many channels).

While interface numbers are shown above, you could have used the names of
the interfaces as shown by CONFIG GET INTERFACES. Remember to use quote
marks around the names!

You may want to remove an ASIOGROUP so that you can use the interfaces
individually. You can do this with:

CONFIG CLEAR ASIOGROUP "name"

Of course this will only work if the group is not currently in use. You
may have to do a CONFIG CLOSE INTERFACE before you can clear the group.

Once you have an interface group defined you use it exactly like you would
use a single interface, for instance:

CONFIG SET INTERFACE "MYGROUP".

4) Timecode generation was broken in the last version or two. This has now been fixed and the timecode generators again work correctly.

5) Timecode following was broken in the last release. If the file position had to seek to follow a skip in the timecode, the track would get marked as not ready to play at the start of the seek and stop playing.
This is now fixed.

6) Setting the start time of the timecode generator while it was running would only randomly affect the generator. It was originally supposed to have no effect until the generator was stopped and restarted, but it didn't quite work that way. It has now been changed to immediately set a new time in the generator.

****

Version 1.0.42
2009-04-02

****

1) Fixed yet another way we could get a dump during shutdown.

2) A nonsensical crash back on version 37 indicates that sometimes ASIO drivers don't work the way the ASIO spec says they should. Put protection code around the attempts to release the ASIO drivers when we no longer need one.

3) Did some minor cleanup on file type determination code.

4) Added log messages to show how many worker threads are being used for the audio processing. Multithreading was added in version 38.

5) Added a checkbox to the interface to disable multithreading on the audio processing. There are reports that some machines click and pop like crazy when multithreading, but work fine normally.

6) Added a log message that shows the version of OS in use.

7) Some optimization code put into version 38 caused ticks when playing back if sample rate conversion was not needed.

8) If a new track was loaded on a playback channel without stopping the previous track, there were cases where the new track could continue to play for various lengths of time. This could also result in various clicks and buzzes, and arcane results like one channel of two playing at twice or three times normal speed. Hopefully all this has now been fixed.

****

Version 1.0.41
2009-03-22

****

1) We could get a tick on the end of a sound file if there were extra hunks on the end of a wave or AIFF file.

2) Fixed a crash that could happen late in exiting the program.

3) SET GROUP G0 TIMECODE UNLOCK would still produce a syntax error due to an incomplete fix in version 39.

****

Version 1.0.40
2009-03-22

****

1) Minor formatting cleanup in an error message that occurs if SMS is unable to send to the TCP socket connected to the controlling application.

2) If SMS got an EWOULDBLOCK on an attempted send to a TCP socket, it would loop retrying the send with no delay. On a single processor system this could have used all of the time needed for a sender to get around to reading from its socket. There is now a 10ms delay between retries.

3) Some locking has been added where selection pointers are being set and cleared in input channels. This should prevent a very rare crash where a channel pointer is deleted from a selection on two threads at once.

4) Fixed some problems with setting up the operating mode and VU meters correctly if CONFIG SET INTERFACE was entered from the command line. This problem would occasionally result in missing VU meters the first time the window was displayed.

5) Changed the memory dump code to make larger dumps. The previous dumps often lacked information necessary to find the problems.

6) CCriticalSection doesn't seem to work as it is defined. Replaced all uses with my own critical section wrapper that does work.

7) Added some interlock logic to prevent attempting two motion commands at the same time on a single channel. The new logic doesn't handle all possible cases, but it handles the cases that can occur commonly with SMA. This prevents some possible very rare crashes in SMS, and also insures that the commands will have the intended result.

8) Found a problem in the thread manager that could cause a crash on shutdown. This is now fixed and SMS shuts down cleanly again.

****

Version 1.0.39
2009-03-21

****

1) A crash that could happen when opening the ASIO interface a second time has been fixed.

2) Unlocking a group of channels from a timecode reader was not working correctly and would give a syntax error. This has been fixed.

3) The previous version broke being able to have the same audio file opened on multiple playback channels at once. This is now fixed.

4) Setting a delay on a playback channel and then stopping it could cause a buzz due to looping the last buffer that was playing. This is now fixed. Note that if you have a delay on a playback channel, the stopping and starting of the channel will seem to be delayed by the delay amount. What is actually happening is that the track is stopping and starting when requested, but the delay is causing the audio to take time to get to the outputs.

****

Version 1.0.38
2009-03-10

****

1) GET GAINMIDI returned incorrect values for input and output channels.

2) Added a CONTROL button to the interface to display the ASIO control panel for the current ASIO driver, if the driver has one. Some ASIO drivers have important configuration settings that can only be accessed through the ASIO control panel interface.

3) Validated operation with AudioScience ASI6464 Cobranet cards and their latest development ASIO driver. When using the latest driver it is necessary the first time the driver is selected to perform the following actions:

   a) Select the AudioScience ASIO driver in the SMS control panel. (If you see more than one AudioScience ASIO driver to choose from, you have an old version of the ASI firmware which will not work correctly. Get a newer download from AudioScience.)

   b) Click the Connect button in the SMS control panel and observe that the ASI driver opens correctly.

   c) Click the Control button in the SMS control panel. The ASI ASIO driver control panel will open.

   d) On the left of the "Adapter and Sample Type" pane of the ASI driver will be a checkbox list of all of the ASI6464 cards in the system. Only one of them will be checked. Check all of them.

   e) On the right of the "Adapter and Sample Type" pane of the ASI driver is a radio button list allowing you to select the sample format that the ASIO driver will use. The 16 bit signed integer button will be selected. As this is a lower-quality sample format, you will want to select one of the other formats. If you are only going to be using the ASI cards with SMS, select the 32 bit floating point option. If you will be using the cards with other applications, select the 32 bit signed integer option.

   f) In the "Buffer Settings" tab you can change the default buffer size for the ASIO driver. This will be set to 2304 samples. This setting has a quite long latency. You can set this to the minimum of 1152 samples, but on a slower machine you may run some chance of having dropped samples. Unless low latency is critical I recommend leaving this option set to the default value.

      Do not change any of the other options in the Buffer Settings tab.

   g) Click OK to save the settings changes. The control panel will close.

   h) VERY IMPORTANT! CLICK THE DISCONNECT BUTTON IN THE

SMS CONTROL PANEL AT THIS POINT. The driver settings have changed drastically, but SMS is still using the old settings. If you do not disconnect from the driver and then reconnect, SMS or the driver will almost certainly crash.

i) Once you have disconnected from the driver you may reconnect and begin using the driver normally. You will not need to visit the ASI control panel again in the future.

4) The history log that appears in a dump file now correctly has timestamps on the log lines.

5) If an ASIO driver callback thread appears to be open after closing the driver, but it terminates normally before we can kill it, we no longer claim that we killed the driver thread.

6) If a driver has Float32 sample format but the buffers are not aligned on 16-byte addresses, SMS would crash. This is now fixed. Sixteen byte alignment is MUCH faster than other possible sample alignments, but having things run at all is faster than crashing and not running.

7) If an attempt was made to open an ASIO driver and the driver failed to load (perhaps because the interface hardware was unplugged) it was possible that SMS would crash. This has been fixed.

8) SET MATRIX no longer includes the signal generators and timecode generators and timecode readers in the matrix setup. Only the live inputs and the playback inputs are included in the setup.

9) The SET MATRIX command has been enhanced. It is now possible to set up only the live inputs or only the playback part of the matrix without also setting the other part. This allows you to leave existing settings for one section while setting the other, or to set different matrix parameters on each section. By default the command will still set both sections.

The syntax now is:

SET MATRIX
IN | INPUT | PB | PLAYBACK | ALL
FULL | DIAGONAL
ON | OFF | GAIN <gain> | GAINDB <gain>

The SET MATRIX ROW command remains unchanged:

SET MATRIX ROW n GAIN|GAINDB channel [-|TO channel] @value...

In the SET MATRIX command the IN/INPUT/PB/PLAYBACK/ALL parameters have been added. Specifying IN or INPUT will limit the matrix settings to the live inputs and the outputs. Specifying PB or PLAYBACK will limit the settings to the playback inputs and the outputs. Specifying ALL, or not specifying any of the new options, will result in setting both the live and playback inputs.

If FULL is specified all crosspoints in the requested sections will be set to the requested gain level. ON and OFF are the same as full gain and zero gain. The input and output channel gains will be set to 1 for any gain other than OFF or 0.

If DIAGONAL is specified all crosspoints in the requested sections will initially be set to 0 gain, and then only the "diagonal" crosspoints will be set to the requested gain. Diagonal crosspoints are those crosspoints with the same input and output channel numbers. The input and output channel gains will be set to 1 for any gain other than OFF or 0.

In all cases the signal generator gains will be set to 0 so that they don't interfere with the overall matrix setup.

10) The GET CHAN <chan> PITCH command now works. Previously it was accidentally not enabled.

11) If you set up a track to loop and also set a non-zero start point for playback, the track playback would seriously mess up when it tried to loop. This has been fixed and loops now work correctly in all known cases.

12) SoundMan-Server now recognizes WAVEFORMATEXTENSIBLE files and most of the possible sample formats. This allows wave files with up to 18 channels of data (or possibly more, but Microsoft only defines 18 channels) and wave files with sample formats larger than 16 bits. The most common sample format is 24 bit PCM samples, but some programs also create 32 bit IEEE floating point samples. There are many other sample formats that are possible in WAVEFORMATEXTENSIBLE, and the more likely ones are handled.

13) SoundMan-Server now recognizes Broadcast Wave Format wave files that can have more than 2GB of audio data. These files use a type of RF64 rather than RIFF. Chained BWF files are now yet handled automatically.

14) SoundMan-Server now plays AIFF files in most non-compressed formats. They are handled exactly like wave files.

15) Much of the internal list and linkage logic has been reworked to reduce the program overhead while playing audio. The reduction probably isn't huge, but it should help on slower machines.

16) When running on a multiprocessor system, SMS will now use all processors to process audio. Previously only a single processor would be used. This should allow considerably more high-overhead processing such as eq and pitch shifting without running out of processor time.

17) Problems with shutdown sequencing when SoundMan-Designer requests that we shut down have been fixed. We should no longer be getting a dump every time the program is closed.

****

Version 1.0.37
2008-12-18

****

1) Updated to new dongle files with some bug fixes.

****

Version 1.0.36
2008-11-30

****

1) A change in the last version broke parsing of input and channel numbers that were entered in lower case.

2) Mixing groups and playback channels in the same command such as PLAY GO,P0 was not working correctly. Only the first item in the list was used.

3) Timecode following was broken in recent versions. This is now fixed and timecode chase again works.

4) A timecode generator chasing another timecode source was very slow in responding to a major jump in the source timecode. This is now fixed.

5) If a parametric filter section is specified as highpass or lowpass and given a gain of exactly 0db, it will now reduce the gain in the filter band relative to the passband. Previously a 0db gain had been accidentally treated as a 'boost' gain, resulting in a gain increase in the filter band.

****

Version 1.0.35
2008-11-05

****

1) The recent changes to the timestamp format on log lines broke the trick of selecting a previous command from the log and having it copied back into the command line. This is now fixed.

2) SET GEQ FREQ 400 GAINDB 2 was setting the wrong band gain.

3) The command to get the graphic eq gain for all bands has been simplified. Previously it was either

    GET CHANNEL <chans> GRAPHICEQ BANDS GAIN
    GET CHANNEL <chans> GRAPHICEQ BANDS GAINDB

Now it is

    GET CHANNEL <chans> GRAPHICEQ GAIN
    GET CHANNEL <chans> GRAPHICEQ GAINDB

Note that the word BANDS is no longer needed. The response format has not been changed, only the command syntax itself.

Also, if you just enter the following with no other parameters,

    GET CHANNEL <chans> GRAPHICEQ

You will now get the band gains in dB. Previously this returned the enable state of the graphic eq for the channel. Note that gains will be shown for the bands even if the eq is disabled for that channel!

The complete list of GET commands for the graphic eq now is:

QUALITY	# GrEqQuality	chanid=number
DELAY	# GrEqDelay	chanid=fpt
ENABLE	# GrEq	chanid=ON|OFF
GAIN	# GrEqGain	chanid=fpt,fpt... 31 times
GAINDB	# GrEqGaindB	chanid=fpt,fpt,.. 31 times
BAND number	#	
FREQ | FREQUENCY	# GrEqBandFreq	chanid=band=fpt
GAIN	# GrEqBandGain	chanid=band=fpt
GAINDB	# GrEqBandGaindB	chanid=band=fpt

4) The timecode generators were not being freed when SMS shut down, resulting in a minor memory leak. This is now fixed.

5) The GET CHAN <chan> GRAPHICSEQ BAND <number> GAIN|GAINDB commands were not returning the band number in the response as they should have.

6) Commands have been added to get the response for both individual parametric eq bands, and for all eq enabled on a channel, including both parametric and graphic eq. The response values are given in dB, where 0dB indicates a "flat" response for the channel. The eq response values do NOT include overall channel gains, only the gain contributions from the eq for the channel.

EQ	# Eq	chanid=ON|OFF
RESPONSE	# EqResp	chanid=fpt,fpt,fpt... 31 times
AT <freq>	# EqFreqResp	chanid=freq=fpt
BAND number		
RESPONSE	# EqBandResp	chanid=band=fpt,fpt,fpt... 31 times
AT <freq>	# EqBandFreqResp	chanid=band=freq=fpt

The command

GET CHANNEL <chans> EQ RESPONSE

Will get the response of all enabled parametric eq bands for the channel, plus the response of the graphic eq if it is enabled. If no eq is enabled, the response will show as flat at all frequencies.

The response is returned as 31 numbers representing dB values for specific frequencies. The frequencies are the same as the 31 band centers for the graphic eq bands. These are logarithmically spaced across the audio frequency spectrum, so will give a good overall impression of the frequency response, but it will not show fine detail from very narrow filter bands.

If it is known than there are very narrow parametric bands in use, the user can get a more accurate overall response by issuing a few commands of the form:

GET CHANNEL <chans> EQ RESPONSE AT <freq>

Where the frequencies are picked to cover the area of the narrow eq band. For instance, this could be used to request the overall response at the center frequency of a bandpass filter to get an accurate value for the bottom of a notch or the top of a bump. Since the 31 band responses are in 1/3rd octave increments they will provide good overall frequency response coverage, and requesting individual responses should only be needed for the center frequency of bandpass filters, or at most a few frequencies around the corner frequencies of the various enabled filters.

Because it can be desirable to show a graphic response of the individual eq bands as well as the overall eq curve, you can get the response for each band individually. In this case the response will show the eq band response even if the band is disabled. It is up to the caller to realize that disabled eq bands do not contribute to the overall channel eq value. These individual band responses do not include any contribution from the graphic eq subsystem. However, you can get the graphic eq response in dB with

GET CHAN <chans> GEQ GAINDB

To get the overall response for an individual band, use a command in the form:

GET CHAN <chans> EQ BAND <band> RESPONSE

This will return the response as 31 numbers, which are the gains in dB at the standard 31 band centers, as described above. The response format (as shown above to the right of the command) is slightly different than described previously, since the band number is included at the front of the response.

Since the eq band might be a narrow notch or bump, you can get the exact response at a specific frequency with

GET CHAN <chans> EQ BAND <band> RESPONSE AT <freq>

This is again very similar to the command described above, except that it returns the response for only the single band requested, and it will show the band response whether or not the band is enabled.

7) Several problems were found and fixed in the parametric eq interrogation commands.

8) If the window is minimized to the taskbar (rather than hidden in the tray) double-clicking the tray icon will now make the window visible. Previously double clicking in this case had no effect and was somewhat frustrating.

9) The error responses returned for bad values of MUTE and SOLO had an incorrect description of the error.

10) On Vista you can't write files to the root directory on the C drive from a program. Therefore the dump files are now written to a directory created in My Documents.

11) If the program is started with the window minimized (as would be the case from SMD) the window will quietly be moved to the tray to clean up the unnecessary icons on the task bar.

12) Commands are now logged in their original case rather than being forced to uppercase before being logged. Previously strings in quotes would have been uppercased in the log records, and the command could then fail if the command was reused from the command history in the control panel. By logging in the original case the case of the strings is preserved.

13) The VU meters were not always being laid out correctly, and would come up as blank a lot of the time.

14) The disk usage meter was unrealistically sensitive and would tend to run very high, even on almost unused disks. It is still deliberately a bit "generous" in its indications, but it is much more realistic now.

15) Found and fixed a very unlikely crash that could occur if you maximized the SMS window at just the wrong time.

16) The ANALOGFIRST parameter was ignored the first time an ASIO port was opened. This is now fixed.

17) The first time you clicked the minimize button on the main window the window would minimize to the tray rather than correctly minimizing to the taskbar. This would only happen once, and then it would work correctly. This has been fixed.

****

Version 1.0.34
2008-10-14

****

1) If a socket error occurs on a send to the socket, the log message is now more explicit about both the text sent and the error returned.

2) The logging code that computed the milliseconds for the current message could have been off by a few milliseconds. This has been fixed.

3) Improved the accuracy of the time function that produces the log timestamps.

4) Eliminated a blank line in the log records for the device channel ordering.

5) If a socket send gets an EWOULDBLOCK response it will now be retried for up to 100ms before it is declared to be an error. This should almost never happen, but can happen for a few milliseconds while SMA is loading a show.

6) When the Close item is selected in the system menu, or the big X in the upper right corner is clicked, the program will now minimize to the tray. The normal Minimize bar (near the big X) will minimize to the normal taskbar location.

   To exit the program, use either the Exit item in the menu on the system tray icon, ir use the new Exit menu item in the system menu.

7) Due to an oversight the code to set up the graphic eq had not been enabled. This is fixed.

8) The GET GRAPHICEQ BANDS GAINDB command was not returning a correct result.

****

Version 1.0.33
2008-10-10

****

1) The GET EQ ENABLED command was erroneously returning solo state, not parametric eq enable state as it should have.

2) If attempting to set the ASIO interface using the Connect button and SoundMan-Assistant is running, a dialog box will warn that this is only a temporary setting, and the right place to set the ASIO interface is in SoundMan-Assistant. If the user clicks on Yes, the SMA configuration dialog will be displayed so that the user can pick the interface there.

3) Parametric eq bands are numbered starting from 1, but the GET EQ responses were showing band numbers starting at 0. This has been fixed.
Also, the GET EQ command expected requested band numbers to start at 0 rather than 1. This has also been fixed.

4) Entering a band number of 0 on SET EQ command erroneously complained that the band number was too large, when it was actually too small. It now just says that the band number is invalid.

5) If you attempt to set frequency or bandwidth for a FLAT parametric filter section you will get an error response, as a flat filter does not have a specific frequency or bandwidth. Previously the parameters were accepted but ignored, leading to confusion about how the eq was working.

6) If you attempt to set the bandwidth for any filter shape except bandpass you will now get an error response. Previously the parameters were accepted but ignored, leading to confusion about how the eq was working.

****

Version 1.0.32
2008-09-12

****

1) Corrected some problems in the code that creates the dump files if something goes wrong.

2) Log messages now have timestamps accurate to about a millisecond, rather than 10-15ms as previously.

3) There is now a date stamp on messages as well as a time stamp. This helps when a log covers several days.

****

Version 1.0.31
2008-08-25

****

1) Log messages now have the time in milliseconds rather than just seconds.

2) SMS log messages can now be captured by the SoundMan-Monitor logging monitor application.

3) A great deal of extra logging of internal events has been added.

4) A new button on the interface will save the log to a file with one click rather than having to copy and paste into Notepad or a mail message.

5) Added code to automatically clear the debug message log if it gets more than about 65,000 messages in it.

6) Fixed up the tab order in the main window to follow a logical progression.

7) Added a top-level unhandled exception filter that will try to take a dump if a crash occurs. If the dumps are sent to me this should result in better support of unusual problems.

8) Fixed a potential problem that could result in a crash when setting up a new audio interface.

9) Fixed a memory leak where we weren't cleaning up the space used by the graphic EQ component.

10) The code in the ASIO callback routine now checks to see if it is using all of the processor time. If so, it begins slowing down so that the UI won't be locked up and require a reboot to regain control of the computer.

    Slowing down will probably result in glitches in the sound, but this is better than locking up the computer. When slowing down like this the computer will be very slow and unresponsive, but with patience it is possible to kill SMS or take some other action to reduce the workload on the computer.

    The most likely reason at the moment for running out of processor time would be either using a matrix size that is too big for the amount of processing power on an older and slower computer, or using too much EQ. When VST plugins are added, they will also be a reason to run out of processing time, as not all VST plugins are efficiently coded.

11) Added a 31 band graphic equalizer to the channels and crosspoints. This can be used as an alternative to the 7-band parametric eq that has always been present.

    The graphic equalizer is implemented using an FFT algorithm. While this allows good control over the overall waveshape and is somewhat less processor overhead than 31 individual parametric filters would be, it has some potential drawbacks, and should be used only when absolutely necessary. The graphic EQ is quite processor hungry and memory hungry, and a small

number of them can make quite a dent in the amount of processor time needed to process the audio.

The graphic EQ has a QUALITY setting of 1 to 8, where 1 is the lowest audio quality and 8 is the highest. The default is 3, which should be sufficient for most uses. The quality setting affects three things:

1. The amount of processor time required for the EQ operation
2. The amount of latency in the audio processing
3. The overall amount of audio distortion that can be caused.

A lower quality uses less processor power. It can also result in aliasing or chorusing distortion, especially at lower frequencies. This distortion is a natural result of how an FFT works: it divides the frequency spectrum into a number of "bins", where each bin contains the same number of frequencies. For instance, a 4-bin FFT would divide the range of 0-48KHz into 0-12000, 12000-24000, 24000-36000, 36000-48000. This FFT could produce severe aliasing an distortion, because all simultaneous tones in the range of 0-12000Hz would be merged into a single output tone. The same would happen in the range of 12000-24000Hz.

This FFT size would be quite low overhead, but the chances of it being very usable are slim. You might think it would be unusable in all cases, but that might not be the case. Consider a single person speaking or singing, with no background sound in the same channel. The human voice generally makes only a single frequency at any instant. So there would be no two frequencies to merge in either of the two important frequency bins, and the resulting output would presumably be the same as the input: no distortion would result.

SoundMan-Server does not use a 4-point FFT as the quality would only rarely be usable. The usable sizes are in the range of 128 to 4096 points. The QUALITY figure is used to select the number of points in the FFT, and also changes some other internal parameters that affect the overall sound quality. The higher the number the more FFT buckets, so the less chance of aliasing distortion.

Before you arbitrarily decide to just set the quality setting to 8 any time you use a graphic EQ, there are two other important things to know:

1. A setting of 8 uses about 64 times as much processor as a setting of 1.
2. The more points in the FFT the longer the latency!

The FFT size can be related directly to the amount of latency in milliseconds it will add to the channel:

FFT size	quality	sample rate	Latency
512	1 2	48000	10.67 ms
1024	3 4	48000	21.33 ms
2048	5 6	48000	42.67 ms
4096	7 8	48000	85.33 ms
512	1 2	44100	11.60 ms
1024	3 4	44100	23.22 ms
2048	5 6	44100	46.44 ms
4096	7 8	44100	92.88 ms

As you can easily see, even a small "QUALITY 1" FFT will add more than 10ms to the audio path. The highest quality FFT processing will add almost a tenth of a second to the audio path. This would not be a good choice to EQ the foldback for a singer or musician. In fact, it would be unusable for any form of live audio, but it could be used on recorded audio. (But then, why didn't you EQ the recording before using it?)

The graphic EQ can be used along with the parametric EQ on the same channel. This should be avoided unless absolutely required, as both the graphic and parametric EQs use a fair amount of processor time. Use the minimum number of parametric bands, as each band added adds more processing time.

There are new channel command parameters to set up the graphic EQ:

```
SET CHAN|CHANNEL channel-spec
        GEQ | GRAPHICEQ
                QUALITY         1..8
                ON|OFF
                BAND int                                        // by band number 1..31
                        GAINDB          fpt
                        =                       fpt             // gaindb
                FREQ | FREQUENCY                // by frequency 20..20000
                        GAINDB          fpt
                        =                       fpt             // gaindb
```

"SET CHANNEL n EQ" and "SET CHANNEL n GEQ" are different. The first sets the parametric EQ sections. The later sets up the graphic EQ.

Each type has its own enable. If the enable is off, the EQ processing is bypassed. As EQ processing adds latency to the channel, you can expect a glitch if you turn the EQ processing on or off. You can keep the same latency by setting the various parametric EQ sections to FLAT or by setting the band gains for the graphic EQ to 0, but this also keeps the same amount of processor overhead for not much purpose.

The QUALITY setting is discussed above. The default QUALITY is 3. You should rarely find a need for a setting above 6, and there are many cases where a setting of 1 or 2 is perfectly adequate.

The graphic EQ works just like the physical graphic EQs that you are used to. There are 31 "gain sliders" numbered 1 to 31 left to right. These are each set at the EIA standard band center frequencies that are used by almost all graphic EQs these days.

Each gain value can be set to anywhere between +24dB and -24dB. The default is 0dB, which does not change the signal level passing through. Note that you can either say "BAND 5 GAINDB 3.6" or "BAND 5 = 3.6" and get the same results. The equal sign will save a little typing.

If you don't remember band numbers but do know the band frequency, you can set the band gains by frequency rather than band number. The frequency you give will be converted to the nearest band number, and then the gain for that band will be set.

Here are the band numbers and center frequencies:

Audio Band#	Nominal Center Frequency Hz	Exact Center Frequency Hz	Passband Hz
1	20	19.95	17.8 - 22.4
2	25	25.12	22.4 - 28.2
3	31.5	31.62	28.2 - 35.5
4	40	39.81	35.5 - 44.7
5	50	50.12	44.7 - 56.2
6	63	63.1	56.2 - 70.8
7	80	79.43	70.8 - 89.1
8	100	100	89.1 - 112
9	125	125.89	112 - 141
10	160	158.49	141 - 178
11	200	199.53	178 - 224
12	250	251.19	224 - 282
13	315	316.23	282 - 355
14	400	398.11	355 - 447
15	500	501.19	447 - 562
16	630	630.96	562 - 708
17	800	794.33	708 - 891
18	1000	1000	891 - 1120
19	1250	1258.9	1120 - 1410
20	1600	1584.9	1410 - 1780
21	2000	1995.3	1780 - 2240
22	2500	2511.9	2240 - 2820
23	3150	3162.3	2820 - 3550
24	4000	3981.1	3550 - 4470
25	5000	5011.9	4470 - 5620
26	6300	6309.6	5620 - 7080
27	8000	7943.3	7080 - 8910
28	10000	10000	8910 - 11200
29	12500	12589.3	11200 - 14100
30	16000	15848.9	14100 - 17800
31	20000	19952.6	17800 - 22400

12) Many new responses for GET CHANNEL info have been added, and several bugs in the existing code have been fixed. The following chart shows the GET CHANNEL options that are now implemented and shows the keyword and result format that will be returned.

In the following table, the part to the left of the # shows the command.

Parts of the commands are indented, and an indented line must be proceeded by the command part on the next outer line. For example, CHAN or CHANNEL must be proceeded by GET (forming GET CHAN or GET CHANNEL). DELAY must be proceeded by CHAN or CHANNEL, which itself must be proceeded by GET, forming GET CHANNEL DELAY.

The part to the right of the # shows the response that will be returned. All responses start with a unique single keyword that is a mixture of upper and lower case. This makes it possible to distinguish easily from an error response or some other line. Since all response starters are unique, if responses are parsed asynchronously it is possible to determine

what each response is about without knowing the command that generated the response.

A GET command can ask about multiple channels at once. Each channel response is in the form of <channel identifier><equal sign><response><space>. The final channel response is followed by a semicolon, which terminates the response line. The entire response, no matter how long, will be on a single "line" with no carriage returns or the like in the response text. There is a carriage return and linefeed after the semicolon at the end of the response.

The channel identifiers are the same as the most compact form used for SET command input. For instance, input channel 0 is I0 and playback channel 10 is P10. Remember that crosspoints have two numbers, so will be of the form X2.3 or PX12.14. There is always exactly one channel identifier immediately followed by an equal sign with no spaces. The response to the query for that channel then immediately follows the equal sign with no intervening spaces. Another channel identifier, equal sign, and response value can then follow the response value for the first channel listed.

Channels are not necessarily listed in the same order in responses that they were given in the original GET command. Do not depend on the order being the same.

Query responses are usually numeric. They can be integer numbers, floating point numbers that contain a decimal point, or a timecode value. None of these items contain spaces. So you can determine the end of the response value for each channel by looking for the first space after the equal sign.

Sometimes the response value is a string value of some form. There are three general cases for this. The first is querying something like the shape of an EQ section. The response is a fixed string that is more descriptive than an arbitrary number would be. This string has a fixed spelling and does not contain spaces, so it is given immediately after the equal sign, and again the end of the response word can be determined by looking for the first space.

The second case of a string response is asking for a channel, group, or other item name. Names are always uppercase and never contain spaces. They may be a mixture of numbers and letters and underscore characters. The first character will always be a letter. Again, names never have spaces in them, so scanning from the equal sign for the first space will find the end of the name field.

The final case of a string response is asking for a file name. Here the file name can contain any characters except a double-quote mark, and can be upper and lower case mixed. The file name can also contain one or more spaces, and very often does. In this case there will be a " mark immediately after the equal sign. The file name will follow this, and will be terminated with another " mark. In this case the file name string must be parsed by looking for the trailing quite mark, and NOT by looking for the next space! There will be a space after the trailing quote mark and then the next channel identifier or the trailing semicolon will show up, in the usual manner.

In the following lines, there are small keywords that indicate the form of the response you should expect. The more common ones are:

chanid the channel identifier, such as P1 or O12 or PX1.1

number	an integer with no decimal point
fpt	a floating-point number containing a decimal point
ON|OFF	either the word ON or the word OFF, in uppercase
none	The word "none" in lowercase. This is used when querying a value and there is no value set.
fpt,fpt,fpt	three floating point number (see above) separated by commas
fpt,...	one or more floating point numbers, all but the last one followed by a comma to separate it from the next one. The last number is NOT followed by a comma.
band	In EQ responses, an integer band number. Note there are TWO equal signs in this form of response for each channel!
timecode	a timecode value of the form 01:02:03:04.05, giving the timecode value to a hundredth of a frame. Timecodes are 30FPS.

```
GET
        CHAN|CHANNEL        channel-spec
                NAME                                    # Name  chanid=name|none
                PORT                                    # Port    chanid=portname
                ROW                                     # Row   chanid=number|none
                GAIN|GAINDB                     # Gain|GaindB    chanid=fpt
                GAINMIDI                                # GainMidi        chanid=number
                PHASEREVERSE                    # PhaseReverse   chanid=ON|OFF
                MUTE                                    # Mute   chanid=ON|OFF
                SOLO                                    # Solo    chanid=ON|OFF

                DELAY                                   # Delay  chanid=ON|OFF
                        ENABLE                          # Delay  chanid=ON|OFF
                        TIME                            # DelayTime       chanid=fpt
                        TC                                      # DelayTC         chanid=timecode
                        FADETIME                        # DelayFade       chanid=fpt
                        FADETC                          # DelayFadeTC    chanid=timecode

                EQ                                              # Eq      chanid=ON|OFF
                        ENABLE                          # Eq      chanid=ON|OFF
                        BANDS                           # EqBands         chanid=number
                        BAND number
                                SHAPE                   # EqBandShape   chanid=band=shape
                                                                #       shape
=FLAT|LP6|LP12|HP6|HP12
                                                                #                       BP
|LS6|LS12|HS6|HS12
                                ENABLE                  # EqBand          chanid=band=ON|OFF
                                PARAMS                          # EqBandParams
chanid=band=freq,gain,bw
                                GAIN                    # EqBandGain      chanid=band=fpt
                                        FADETIME        # EqBandGainFade          chanid=band=fpt
                                        FADETC          # EqBandGainFadeTC
chanid=band=timecode
                                GAINDB                  # EqBandGaindB chanid=band=fpt
                                        FADETIME        # EqBandGaindBFade        chanid=band=fpt
                                        FADETC          # EqBandGaindBFadeTC
chanid=band=timecode
                                BANDWIDTH|BW    # EqBandBW      chanid=band=fpt
                                        FADETIME        # EqBandBWFade            chanid=band=fpt
                                        FADETC          # EqBandBWFadeTC
chanid=band=timecode
                                FREQ|FREQUENCY  # EqBandFreq      chanid=band=fpt
                                        FADETIME        # EqBandFreqFade          chanid=band=fpt
                                        FADETC          # EqBandFreqFadeTC
chanid=band=timecode

                GEQ|GRAPHICEQ
                        ENABLE                          # GrEq   chanid=ON|OFF
                        BAND number    #
                                GAIN                    # GrEqBandGain chanid=band=fpt
                                GAINDB                  # GrEqBandGaindB          chanid=band=fpt
                        BANDS
                                GAIN                    # GrEqGain        chanid=fpt,fpt... 31 times
                                GAINDB                  # GrEqGaindB    chanid=fpt,fpt,.. 31 times
```

```
PITCH                                   # Pitch   chanid=ON|OFF
        QUALITY                         # PitchQuality     chanid=number
        DELAY                           # PitchShiftDelay  chanid=fpt
        ENABLE                          # Pitch   chanid=ON|OFF
        FFTSIZE                         # PitchFftSize     chanid=number
        OVERLAP                         # PitchOverlap     chanid=number
        SHIFT                           # PitchShift       chanid=fpt
                FADETIME                # PitchShiftFadeTime       chanid=fpt
                FADETC                  # PitchShiftFadeTC         chanid=timecode

TRACK
        FILE                            # TrackFile        chanid="name"
        STATUS                  # TrackStatus      chanid=PLAY|STOP
        POSITION                        # Position         chanid=fpt
        TIME                            # TrackTime        chanid=fpt
        LENGTH                          # TrackLength      chanid=fpt
        SPEED                           # TrackSpeed       chanid=fpt
        PITCH                           # TrackPitch       chanid=fpt
        START                   #
                TIME                    # TrackStart       chanid=fpt
                TC                              # TrackStartTC    chanid=timecode
        STOP                    #
                TIME                    # TrackStop        chanid=fpt
                TC                              # TrackStopTC     chanid=timecode
        REPEAT                  #
                TIME                    # TrackRepeat      chanid=fpt
                TC                              # TrackRepeatTC chanid=timecode
```

13) Changed the first log message of "SoundMan-Server starting" to include the current version number.

14) If a sound file fails to open, the Windows error message will be included in the log message.

15) The Configuration Limits log line now shows the maximum licensed sample rate as well as the chosen sample rate.

16) A bug in input converter setup could have caused some wave file channels to not play when they should have,

17) Fixed a number of minor inconsistencies in the menus between SMA and SMS. Also insured that the version number will show up in the title bar of the main dialog at all times.

****

Version 1.0.30
2008-06-27

****

1) A small problem in delay time calculations that could make small delay errors is fixed.

2) A recent change was causing input delay values to be doubled incorrectly.

3) Changed a compile option to put more debug data in the code file. This should help if it is necessary to look at crash dumps.

4) Added syntax to allow setting the channel delay in samples as well as fractional seconds or a timecode value. Also, if the time value is entered as a number without "TIME" in front of it, you can now also enter any of the other delay parameters. Previously all that was allowed in this format was "DELAY <time>".

   The new syntax now is:

```
SET CHAN|CHANNEL channel-spec
        DELAY
                <number> [SAMPLES]
                TIME fpt [SAMPLES]
                TC                  timecode
                FADETIME fpt
                FADETC   fadetime
                ON|OFF
```

5) SMS will no longer completely lock up the system if the audio processing takes more time than there is available. The processing routines will check if the user interface is able to run at all, and if not, will give a few milliseconds per second to the UI processing. This may very well cause glitches in the audio; however, glitches are very likely anyway when all of the processor time is in use.

   A balloon popup will be displayed by the tray icon indicating that the processing is running overtime. On a single processor system the mouse and keyboard will be very sluggish, but enough time will be left that it will be possible to stop SMS (or change some parameters) without having to reboot the system.

6) It is now possible to do a pitch shift on any input or playback channel. This is a processing pitch shift on the audio stream, NOT changing the playback speed to achieve a pitch shift. On playback channels the channel pitch shift can be used with an inverse track speed or pitch shift to achieve a change in playback speed without changing the output pitch.

   The pitch shifting routines use a fair amount of processor, so the number of channels they can be used on is limited. Also, they do not guarantee identical phase tracking over short durations on multiple channels, so it is possible to get some stereo image shifting when pitch shifting a multi-channel audio stream.

Several processing quality settings are possible. The standard setting should be good for many uses, but will fail miserably with dense rock music that is highly compressed. It can work quite well on vocal and folk music.

Higher quality settings have two drawbacks: they can use MUCH more processor power, and they can have very long processing delays. The standard level of processing results in about a 20ms delay. Delays up to 170ms are possible at the high quality settings. It can be possible to get acceptable results with some audio sources using a low quality setting that will have a 10ms or even a 5ms input to output delay. Whether this can be done will depend entirely on the nature of the source audio material, and of course how critical you are of the results.

Pitch can be shifted plus or minus one octave, and the shift value is entered in semitones, possibly with fractional semitone values.

SET CHAN <channels> PITCH [ON | OFF] <semitones> [FADETIME <time>]

7) A bug in the command parser that could cause it to lock up on some kinds of invalid input syntax has been fixed.

8) The SET GENERATOR syntax has been enhanced to allow sweep parameters of ON and OFF as well as PAUSE, STOP, and RESUME. ON and RESUME are the same and will resume the sweep if it is paused. STOP, PAUSE and OFF will stop the sweep if it is currently running, holding the current frequency.

The syntax for the generator is now:

```
SET GEN | GENERATOR [channel]
        SHAPE  SINE|SQUARE|TRIANGLE|WHITE|PINK
        FREQ|FREQUENCY       fpt 5..24000
        SWEEP PAUSE|STOP|RESUME|ON|OFF| minf maxf time
```

9) Setting the frequency of the signal generator will pause any current sweep and set the new frequency as a constant value.

10) Get track position now will correctly return an integer value in samples rather than a floating point value.

11) Chasing a timecode that jumped around or had a large offset into the files tracking the timecode could behave poorly. This has been greatly improved.

****

Version 1.0.29
2008-06-07

****

1) Moved to a new version of the ASIO SDK that makes it easier to handle multiple ASIO interfaces at once. Still need to do more to actually open multiple interfaces at once.

2) Fixed up the code that closes ASIO drivers to handle stupid drivers like the Maya driver that do not terminate their callback thread during ASIOExit. Previously this could result in random crashes due to the driver data disappearing out from under the still-running callback thread,

3) Found a problem with the callback routines that with some drivers could sometimes cause a crash. Fixed it.

4) Fixed some static variables in the pink noise generator so that multiple pink noise generators can be used at the same time without interference.

5) SET GROUP parsing was incorrect and would not allow more parameters after the NAME parameter. This is fixed.

6) An uninitialized variable could make a valid SET GROUP command fail.

7) Channel groups were not being constructed properly unless there was a timecode reader attached to the group. Now they are.

8) A new GET command has been added: CONFIG GET CHANNELINFO. This command will return a multiline response that will display the ASIO description string for all of the input channels, followed by all of the output channels. This can be useful in figuring out which channel number corresponds to which physical input or output on devices with multiple different interfaces.

9) Some ASIO interfaces mix up the channel order. For instance, if you have two M-Audio Delta 1010 interfaces, they will work together. However, the analog channels on the second interface will be separated from the analog channels on the first interface by the sp/dif channels and the mixer return channels. If you have a 16 channel license, you will not be able to use all 16 channels.

   To get around this there is a new modifier on CONFIG SET INTERFACE. If you add ANALOGFIRST to the parameters, the descriptions of all of the channels will be examined, and any channel that has the word "analog" in its description will be moved before any other channels. Other than moving the analog channels before other channels, the channel order is preserved. In the case of the Delta 1010s, the channels will be ordered with the 16 analog channel first, followed by the sp/dif channels for the first interface and then the sp/dif channels for the second interface. This can also be useful with a MOTU 324 or 424 card and multiple 2408 interface modules to get the analog channels for all interfaces on the low channels.

   Note that this trick will only work if the description string for the channel contains "analog" in any case, including "Analog" and "ANALOG".

If there are no channels that declare themselves to be analog the channel order will not be changed.

The form for CONFIG SET INTERFACE is now:

```
//      CONFIG|CONFIGURATION
//              SET
//                      INTERFACE       n|"name"
//                              SAMPLERATE      n
//                              INPUTS          n
//                              OUTPUTS                 n
//                              PLAYBACKS       n
//                              ANALOGFIRST
```

10) To see the description strings the hardware manufacturer has provided for the audio channels (and thus determine if ANALOGFIRST will work) you can now do CONFIG GET CHANNELINFO once the interface is open. This will produce a multi-line response, listing the description strings for all of the input and output channels, in order. Note that all of the channels on the interface will be shown, even if the license limits you to less than all of the available channels.

11) To see the description strings for the channels actually in use on the interface (and to verify that ANALOGFIRST did what you wanted it to do), you can use CONFIG GET CHANNELMAP. This will produce a multi-line response that will show the internal channel number, the interface channel number, and the description string for that channel. The output is limited to the number of channels actually in use, which may be less than the physical number of channels available on the interface.

12) If the dongle is set to allow any operating mode, a CONFIG SET INTERFACE with no mode parameter will now open the interface in NORMAL mode. Previously it defaulted to AUDIOBOX mode, and you would have had to override that to get normal mode.

13) The CONFIG SET INTERFACE command with no channel count modifiers will now attempt to open the maximum licensed number of channels by default. Previously it defaulted to 16 channels.

14) Replaced popup message about "initialization failed" with a balloon message on the tray icon. This will no longer require clicking on OK to continue working. Also eliminated a possible popup message when setting interface parameters if the set failed. This has been replaced with a log message and a balloon message.

15) Multiline responses from various commands will no longer have an extraneous blank line following the end of the response.

16) Multiline responses are now formatted correctly in the history log.

17) If there is no dongle present there was an extraneous blank line in the history log on CONFIG SET INTERFACE. This is now fixed.

18) The DELAY parameter for channels now accepts the following syntax:
DELAY <time> FADETIME <fadetime>;
Previously "DELAY <time>" was only permitted if none of the other delay

parameters were present; to fade a delay you had to say
DELAY TIME <time> FADETIME <fadetime>;

19) When doing multiple slow delay fades with large delay values on an output channel, cumulative roundoff error could eventually end up with an incorrect delay value, sometimes producing audible spatter. This has been fixed.

20) All input, output, and crosspoint channels now do delay fades as temporary pitch shifts while the fade is running. Previously only output channels did this. This replaces the delay fade form that skipped and jumped to try to not do a pitch shift on the fades. That form of fade would still sound like a pitch shift under many conditions, and additionally had severe noise problems under many conditions.

21) Playback pitch change was broken in recent versions and was not setting the correct semitone value requested.

22) Playback pitch and speed changes had gotten all screwed up in recent releases, and are now fixed. Playback pitch and speed are separate controls for the same thing: the speed the track will play at. Both playback pitch and playback speed will change how fast the track will play. The only difference between them is "speed" is a multiplier of track speed, where 1 is normal track speed, 2 is double speed, and 0.5 is half speed. Pitch on the other hand is speed expressed as musical pitch in semitones. So a pitch of 0 is normal speed, a pitch of 12 is double speed, and a pitch of -12 is half speed.

Speed and pitch for playback are multiplicative. If you double the speed and take the pitch down an octave, the track will play at normal speed, because 2 times 0.5 = 1.

Speed and pitch changes to a playing track are ephemeral. They will be remembered as long as the track is playing. But when the track stops playing and is restarted, any speed or pitch changes made during the last playback are forgotten.

Sometimes it is desirable to set a speed or pitch variation to be used on the entire playback of the track, and you want to keep that modification even if the track is stopped and restarted. There is a special way to do this. If the speed or pitch parameter is entered on the same SET command with the file name, then the speed or pitch change will be remembered and reused every time the track is played. This speed variation can be overridden with other speed and pitch commands while the track is playing. But as mentioned above, these extra changes to the playback speed will be lost if the track is stopped and restarted. Only the speed or pitch change entered on the initial SET command that loaded the track file will be remembered.

23) CONFIG GET INPUTS, CONFIG GET OUTPUTS, and CONFIG GET PLAYBACKS are now valid before the ASIO interface is opened. When requested before the interface is opened, these commands will return the licensed number of channels possible. After the interface is opened these commands will return the actual number of each type of channel available.

24) Included new dongle files with more error recovery abilities.

25) Increased text fields for dongle info from 50 characters to 100, since the dongle will allow at least 64 characters.

****
Version 1.0.28

****

1) SMS will now show a tray icon rather than a normal program taskbar icon when it is minimized. You can use the tray icon to display the window, show the About Box, or exit the program.

2) If we fail to connect to an interface, and the driver name starts with the string "MOTU", we will not do a messagebox about the driver failure, since the
MOTU drivers already do this themselves. We don't need two annoying message boxes when one will do.

3) In earlier versions if we failed to connect to the driver we could insome cases crash shortly after. This has been fixed.

4) If the driver fails to open, a balloon tooltip will show over the icon in the
lower right corner of the desktop. This will give the operator a hint as to why sound might not be coming out even though SMD seems to be running cues.

5) Fixed a small memory leak that could sometimes occur under very unusual conditions.

6) The positive gain limit for a submaster was 20dB,consistent with all other gain controls. However SMA can send positive submaster gains up to 47.25 db. Changed the limit for positive submaster gain to 50db. The range is now -160 to +50. In AB mode or CC mode the overall gain of the control point will still be limited to +0db maximum.

****
Version 1.0.27

****

This is a major new version of SoundMan-Server which reads and generates SMPTE time code and has a large number of timecode manipulation, chase and lock capabilities.

Here is the summary of this release, provided by Loren:

This version of SMS can both generate and read SMPTE timecode, and can lock any number of playback channels to the input timecode. You can also lock the timecode generators to incoming timecode from a reader, so you can either regenerate timecode or you can generate timecode of a different form locked to the input timecode (for instance, generate 24 fps timecode from 30 fps timecode.

Note that in this version the reader and generator deal with SMPTE, *** NOT *** WITH MTC!

SMPTE readers are "fake output channels". There are two of them, on channels O1000 and O1001. You can route any input channel or for that matter playback channel to either of the readers. You should never route more than ONE source to a given reader if you want it to decode correctly

SMPTE generators are "fake playback channels". This lets you start and stop them in sychronism to other playback channels, and to set speed and pitch and starting time using "track" parameters, just as you might do if you were playing back a timecode track recording.

There are two SMPTE generators, on channels P1000 and P1001.

Playing back multiple tracks in sync to timecode requires the use of a GROUP. There are a number of group channels with names starting from GROUP 0 or G0. The group is the ringmaster that receives incoming timecode from a timecode reader and makes sure that all of the playback channels in the group stay in sync.

To make this work you first have to declare a group that has a number of playback channels in the group, and which also has a SMPTE reader. You also specify the timecode lock mode and starting timecode value for the tracks. All tracks in a group need to start at the same timcode value, and tracks locked to timecode position cannot be set to loop. If the track looped there would be multiple possible timecodes for each position in the track. This is not currently allowed.

There are two ways you can lock tracks to timecode. You can lock them by pitch/speed, or you can lock them by time.

Sometimes all you need is to make sure that your playback channels will play at the same speed as some external device, like a video projector, but you can start and stop the sound manually. If this is what you want, you want to lock to timecode speed. This lets you start and stop tracks manually, and the tracks can loop. However, they will faithfully follow the incoming SMPTE playback rate.

More commonly you want to lock the playback tracks to a constant time position. When you do this you can set all of the tracks to be locked (in the group declaration) and start the tracks. If timecode is not present, or it is before the track starting time, the tracks will not play. Once the timecode appears and is stable and within the track range the tracks will play to timecode. You can manually stop the tracks even if timecode is still playing, and they will stop. When you start them again there will be a short delay while they seek to the current position for the current timecode value.

Audio will only follow timecode that is running forward. If the timecode stops, or starts running backward, or runs at a constant frame number, the audio will not play. Once the timecode is running forward and is within the track time range the tracks will play IF they have previously received a PLAY command.

When playing tracks to timecode, you do not send the Play command to the individual tracks as you would normally do. Instead, you tell the group to play, for instance, PLAY G0.

The commands necessary to play to timecode and all specific new commands are described below.

1) There are now two signal generators. Previously there was a single generator. Both signal generators are identical, and can be used to generate two different tones or other wave shapes routed to different outputs. The SET GENERATOR and GET GENERATOR commands have been enhanced to allow you to specify the generator of interest. The syntax now is:

SET {GEN | GENERATOR} [channel number]

SHAPE SINE|SQUARE|TRIANGLE|WHITE|PINK
FREQ|FREQUENCY fpt 5..24000
SWEEP PAUSE|STOP|RESUME|min max time

GET {GEN | GENERATOR} [channel]

SHAPE
FREQ|FREQUENCY
SWEEP [MIN|MAX|TIME]

This is identical to the previous syntax, with the addition of the optional generator channel number following GEN or GENERATOR. If the channel number is not given it will default to 1000, which is the first generator. If the channel number is given, it must be either I1000 or I1001, as those are the two signal generator channels.

2) There are now two timecode generators on channels P1000 and P1001. The timecode generators can be used independently to generate two different timecode streams. The timecode streams can have different time bases, or can have different timecode types. For instance, you could generate SMPTE at 30 frames/second from one generator and 24 frames/second from the second generator if you needed to lock a projector and a lighting console to the same time, but they required different frame rates.

The timecode generators appear to be playback channels. Since they are playback channels they can be started and stopped with PLAY and STOP commands, just like real playback channels. This lets you generate timecode in synchronism with audio playback. The timecode generators can start, stop, pause, loop, and change speed just like playback channels, so can track all movement actions identically to playback channels.

The timecode generators can generate many different forms of timecode. They can also generate pure sine waves, so can be used as an additional two signal generators if this is desired. By default the generators generate full output level. The command syntax for the timecode generators is similar to the setup for normal signal generators:

SET {TCGEN | TCGENERATOR} [channel number]

```
   FREQ|FREQUENCY fpt 5..24000
   CODE
      SINE
      SMPTE30
      SMPTE30D | SMPTE30DROP
      SMPTE25
      SMPTE24
      IRIG A  [INV|INVERTED]
      IRIG AMOD [INV|INVERTED]
      IRIG B  [INV|INVERTED]
      IRIG BMOD [INV|INVERTED]
   START    [TIME|TC] <start time> [AT [TIME|TC] <offset>]
   DATE     yy/mm/dd
   DAY      nnn
   USERBITS nnnnnnnn
   SPEED    fpt number
   RUNNING  YES | NO | ON | OFF
```

GET {TCGEN | TCGENERATOR} [channel]

```
   FREQ|FREQUENCY
   START {TIME | TC}
   CODE
   TIME
   DATE
   DAY
   USERBITS
   SPEED
   RUNNING
```

The frequency should only be set if you are generating a sine wave. When you set any other code type the frequency is set automatically to the correct rate for that timecode type. It is possible though to manually set the frequency after setting the timecode type. This will probably cause the timecode to run off of the correct frequency, but there may be times when that is useful. The timecode type generated are:

```
SINE         A plain sine wave
SMPTE30      30fps non-drop SMPTE
SMPTE30D     29.97fps drop frame timecode
SMPTE30DROP  29.97fps drop frame timecode
SMPTE25      25fps SMPTE timecode
SMPTE24      24fps SMPTE timecode
IRIG A       1000Hz IRIG-A pulsed timecode
IRIG AMOD    10KHz IRIG-A sinewave modulated timecode
IRIG B       100Hz  IRIG-B pulsed timecode
IRIG BMOD    1000Hz IRIG-B sinewave modulated timecode
```

SMPTE timecodes are most commonly used for synchronization between show equipment of various forms. IRIG is an older timecode form that is very good for long-range synchronization, between hundreds of feet and for some codes, tens of thousands of miles. It tends to be more resistant to noise and distortion

than SMPTE timecode. However, it is polarity sensitive to decode correctly. Since the output polarity of an audio channel is not necessarily predictable, you can generate the IRIG codes either positive or inverted.

The TIME field is used to set or get the starting (or current)generation time. The time will start from zero by default. If the playback channel for the generator has a TRACK START TIME specified, the TIME field will be the timecode value corresponding to that relative position. If no track start time is given, the TIME timecode is the time generated at the front of the track.

The DATE field is used to set or get a date field. This will be placed into the USERBITS field for SMPTE. The field will be validated to insure that it is a valid date. If the DATE and USERBITS parameter are both used and DATE occurs second, it will override the USERBITS value. The date value is initialized to zero by default, and is not automatically rolled over.

The DAY field takes a Julian day number between 1 and 366. This will be inserted into the Julian day field for IRIG timecode. This value is automatically incremented when the time rolls over from 23:59:59 to 00:00:00. It will also automatically roll over from 366 to 1, but not from 365 to 1.

The USERBITS field will place an arbitrary hexadecimal value into the SMPTE timecode user bits field. This will override any value set with the DATE parameter if it occurs after the DATE parameter. For IRIG timecode, the low 27 bits of the userbits value will be placed in the "control bits" field in the timecode value.

The SPEED value can be used to tune timecode generation to match some external source that is running at the wrong speed. Normally pitch = 1.0, which is exact speed. A smaller value will slow down timecode generation and a larger value will speed it up. It is generally more practical to use the TRACK SPEED parameter on the generator channel than to use the SPEED parameter on the generator command.

The RUNNING value will start or stop the generator. The generator is stopped initially, and can be started from the generator command or from a PLAY or RESUME command, possibly along with other playback channels that it will then track.

3) The annoying flutter of the VU meters with a constant signal level has been corrected.

4) The PLAY command will now start all tracks at the defined start point for each track. Previously it always erroneously started from the first sample of the wave file, regardless of the defined starting position.

5) Fixed a very rare crash that could happen if you were positioning into a wave file and closed the file at the same time.

6) There are now two timecode readers on channels OUT 1000 and OUT 1001. You can route signal from any input (or playback) channel to either of the timecode readers. Typically you would have an incoming SMPTE (or IRIG) timecode on an input channel, for instance from an external SMPTE generator or other source. However, you might have a click track or other canned show where the SMPTE source was a recorded track, so you can also route from a playback channel to the timecode readers. You should never route more than one source channel to any single reader. It is not illegal to route multiple channels, but doing so will degrade the timecode decoding if there is noise on the other channels.

Obviously if more than one timecode source is fed to a single timecode reader at the same time, chaos will result, and in all probability the correct timecode values will not be read. The current timecode read by the first reader (channel O1000) is displayed on the debug monitor display for SoundMan-Server. The timecode from the second reader is not displayed. There are currently no options to set in the timecode readers. They decode the incoming timecode data, determine the general format, and then produce an output that can be used to synchronize playback channels and the timecode generators. NEVER lock a playback channel or timecode generator to a timecode reader if that channel is supplying the timecode to the reader! This will result in positive feedback, and the timecode will very quickly run off the tracks.

7) You can now lock a group of playback channels to a timecode reader. Note that the timecode generators are also considered to be playback channels, so you can lock the timecode generators to the timecode readers. You might do this to regenerate timecode, or to generate timecode of a different format slaved to incoming timecode. For instance, if you have 30 fps SMPTE coming in and you need to slave a projector to the timecode, you could generate 24fps SMPTE in one of the generators and lock that generator to the reader following the 30fps timecode. You lock the playback channels to a reader indirectly through one of the group channels. The group channel insures that all of the playback channels in the group will remain in sample-accurate sync while following timecode.

If you locked individual channels to the timecode reader, the channels would follow the timecode within a fraction of one frame, but sample accuracy between the channels could not be guaranteed. The SET GROUP syntax has been extended to deal with timecode locking. The syntax now is:

```
SET { G | GROUP } < nn >
            nn = group number 0 to 127 or group name
   NAME < alphanumeric group name >
   { CHANS | CHANNELS } < channel_list >
   CLEAR
      remove all channels
   { TC | TIMECODE }
      READER { channel number }
      LOCK
         SPEED
         { TIME | TC } < timecode > [ AT { TIME | TC } value ]
      UNLOCK
```

Setting a group sets group properties. Group properties include the list of channels in the group, an arbitrary name for the group, and various
timecode-following parameters.

By grouping channels you can send many commands to the group that you would normally send to a list of channels. In that sense they are somewhat like a VCA on an analog mixer.

You can also group playback channels that will play simultaneously to timecode. Here the group is more important, since it insures that all of the channels will remain exactly in sync as they follow the timecode variations.

Examples

```
SET GROUP G0 NAME BACKING_TRACKS CHANS P0-P7
SET G BACKING_TRACKS TC READER 1000 LOCK TC 01:00:00:00
PLAY GROUP BACKING_TRACKS
```

```
SET GROUP G0 CHANS P0-P7
SET G0 TC READER 1000 LOCK TC 01:00:00:00
PLAY G0
```

Either of the two above sets do exactly the same thing.

Another typical example might be:

```
SET GROUP G0 CHANS P0-3 TC READER O1000 LOCK TC 1:0:0:0 AT 0
```

There are two ways you can lock to timecode: you can just follow the speed or rate at which the timecode is running, or you can lock to the actual timecode position. Sometimes it is sufficient to insure that a

playback started at some random time will remain in sync with some other device, for instance a video playback. If the audio runs at the same speed as the video source the lock will be maintained over extended periods of time. If this is all that is required, you can do TC LOCK SPEED to insure that the playback and the timecode will run at the same rate. Note that the audio will NOT follow the timecode if the timecode value jumps, since only the speed of the timecode is being followed. If the timecode slows down the audio will slow down to match.

More commonly it is necessary to lock the audio playback to a particular time relationship in the incoming timecode stream. If you do this and the audio channel is in playback mode, the audio will start and stop if the incoming timecode stops. If the timecode jumps, the audio will also jump to follow the timecode. If the timecode is arriving continuously but is showing the same frame value over and over, the audio will be frozen at the current time position. It will begin playing the instant the timecode begins moving. Audio will not follow timecode that is running backwards. For that matter the timecode reader will not follow backwards timecode. The timecode must be running in the forward direction to be recognized. (Stationary timecode must be presented with the bits in tor forward direction to be recognized.)

Audio will speed up or slow down as necessary to remain as close to the current timecode value as possible. However, timecode, especially when generated by a mechanical source like a projector, can have a lot of wow and flutter. The timecode reader aggressively filters the timecode values to eliminate flutter and to reduce wow to the lowest possible levels consistent with remaining in sync with the timecode. Note that the audio pitch will change as the timecode speed changes. Thus if the timecode is shuttled forward, you would expect the audio playback to increase in frequency during the shuttle movement. (The audio will not currently follow a backwards shuttle.)

If the timecode jumps, the audio will jump to follow it. Depending on the number of tracks being played and various other conditions, there may be a short silence while the audio files seek to the new position and re-lock to the timecode. Timecode jumps either forward or backward will be followed, as long as the timecode bits continue to run forward. When locking to timecode, you need to specify a timecode value for some position in the audio track.

Typically you would specify the timecode value for the front of the track. However, it might be that the important timecode position is 12.71 seconds into the track. So that you don't have to convert 12.71 to frames at the current timecode rate and subtract that from the desired starting timecode, the group syntax lets you specify both the locking timecode value and where it appears in the track. In the example given above, the first sample of the track file occurs at one hour of timecode. However you could have specified LOCK TC 1:0:0:0 AT TIME 12.71; This would have made the one hour mark occur exactly 12.71 seconds into the track file.

8) The various bar graph displays would freeze when the interface was closed. This is now fixed and they update continuously.

9) The IO usage bar graph no longer hangs non-zero when nothing is playing.

10) There are now two timecode display lines, one for each timecode reader.

11) Commands have been implemented to get the current speed multiplier and pitch multiplier from a playback channel. These commands have the format

GET {CHAN|CHANNEL} <channel-list> TRACK SPEED
GET {CHAN|CHANNEL} <channel-list> TRACK PITCH

The full implemented syntax for GET TRACK is

GET {CHAN|CHANNEL} <channel-list>

    TRACK FILE   # TRACK FILE "name"

```
START|STOP|REPEAT         # TRACK WORD ON|OFF NOTIFY ON|OFF
TIME|TC                   # TRACK WORD TIME fpt | TC tc
LOOP                      # TRACK WORD LOOP ON|OFF NOTIFY ON|OFF
START|STOP|REPEAT TIME|TC  # TRACK START|STOP TIME fpt | TC tc
SPEED                     # current speed multiplier
PITCH                     # current pitch multiplier
POSITION                  # current position in samples
TIME                      # current position in seconds
LENGTH                    # track length in seconds
STATUS                    # PLAY or STOP
```

****
Version 1.0.26

1) Changed the code that reads temporary dongle files to be more reliable.

****
Version 1.0.25

1) You can now copy/paste from the Response line in the control panel.

2) The window position is now saved in the registry rather than an ini file.

3) Added the dongle serial number to the About Box contents.

4) Changed copyright date to include 2008.

5) You can now copy and paste from the info panel in the About Box.

6) The operating mode indicator on the control panel is now reliable.

7) Changed the warning message when we can't set the desired sample rate from a popup to a log message. This keeps from screwing up programs that open SMS remotely.

8) All single-line responses except OK or ERROR should now be terminated by a semicolon to make parsing easier. The caller receiving a response can tell if it is single-line or multi-line by making some simple checks:
a) If the first word is OK or ERROR it is a single-line response.
b) If the response line ends with a semicolon it is a single-line response.
c) In all other cases it is the first line of a multi-line response and the complete response will end with a line containing a single ".\r\n" in the first character position. Asynchronous responses (ones that are not the direct result of a preceding command being issued) are not yet implemented. However when they are, the first character of the first line of an async response will be an "@" character to flag the line as async. If it is a multi-line response only the first line will be flagged with an @ sign. It is expected that all async responses will be single line responses, but the program receiving one should check the rules given above to determine if the response is a single-line async response or the first line of a multi-line async response. All multi-line responses are guaranteed to be transmitted as a group. No other single or multi-line response will ever be mixed with another multi- line response.

9) If you try to open an audio file and request a track number that exceeds the number of tracks in the file, you now get an error message that clearly states the problem. Previously you got a generic "could not open file" error message, which was confusing since the file was in fact opened.

10) When closing the ASIO interface, we now lower the priority on thecallback thread first. This should prevent broken drivers locking up the entire system if they loop during the close attempt. However, SMS itself will probably still get locked up in this case and have to be killed and restarted. Better SMS than the entire system.

11) Added an extra half second delay in the process of closing an ASIO driver. The Maya driver takes a very long time to shut down the callback thread (which should have been complete when ASIOStop returned) and the callback thread would then die after the buffers had been freed. The extra delay gives the callback thread time to stop, and then close completes cleanly.

12) You can now give start, stop, loop, and repeat positions in samples as well as time. This can be handy for controlling programs that have accurate sample position info in the files being played. The sample positions are in terms of file sample rate, which may be different than the interface sample rate. The syntax for SET TRACK has been changed to add this new feature. The changed areas are:

SET CHAN <channels> TRACK

START
    SAMPLES <samplenumber>

TIME <time in seconds, floating point>
TC <time in timecode value>
<time in seconds, floating point>

STOP
SAMPLES <samplenumber>
TIME <time in seconds, floating point>
TC <time in timecode value>
<time in seconds, floating point>

REPEAT
SAMPLES <samplenumber>
TIME <time in seconds, floating point>
TC <time in timecode value>
<time in seconds, floating point>

LOOP FROM
SAMPLES <samplenumber>
TIME <time in seconds, floating point>
TC <time in timecode value>
<time in seconds, floating point>

LOOP TO
SAMPLES <samplenumber>
TIME <time in seconds, floating point>
TC <time in timecode value>
<time in seconds, floating point>

Note that either LOOP or REPEAT should be used for looping, but not both.

13) A "REPEAT <channels> NOW" command will now correctly loop back to the start position for the tracks rather than stopping the tracks.

14) Get channel gain for a crosspoint channel returned an incorrect channel number.

15) Get channel gaindb failed to indicate in the response that the value was in dB. The lead word is now correctly "GaindB" rather than "Gain" (which indicates a linear gain value).

16) Getting the channel gain in dB for a channel with zero gain returned an invalid gain number. It now arbitrarily returns a value of -144 for zero gain.

17) You can now GET the attributes of the signal generator. GET {GEN|GENERATOR} FREQ|FREQUENCY SHAPE SWEEP [MIN|MAX|TIME] FREQ or FREQUENCY returns the generator frequency. SHAPE returns the generator waveshape. This can be one of Sine Square Triangle WhiteNoise PinkNoise PinkSweep Sine, Square, and Triangle are pure tone waveshapes. If sweep is disabled, the generator makes a constant frequency. If normal sweep is enabled, the tone is swept from the minimum frequency to the maximum frequency in the given time, and then swept back to the minimum frequency in the same amount of time. The amplitude of the wave is constant across all swept frequencies. WhiteNoise and PinkNoise are random noise generator functions. WhiteNoise has equal amplitude at all frequencies. PinkNoise decreases in amplitude with frequency. PinkSweep is a swept sine wave that has an amplitude envelope matching the pink noise emvelope. This can be used with a traditional spectrum analyzer to make various passband measurements The PinkNoise function is more appropriate when using an RTA. SWEEP returns one of four values, depending on what follows the word SWEEP. SWEEP MIN returns the minimum sweep frequency SWEEP MAX returns the maximum sweep frequency SWEEP TIME returns the time in seconds for the sweep. SWEEP with nothing following it returns On or Off.

18) If a "config set interface" command is issued that has exactly the same parameters as the current open interface, the interface will no longer be closed and reopened, as there is no need to do that.

****
Version 1.0.24

****

1) Changed the dongle stuff to use the static library, so it is no longer necessary to have Rockey2.dll available.

2) Fixed config stuff to correctly allow more than 16 playback channels in
modes other than AB mode.

3) Implemented a commandline lockout feature. CONFIG SET COMMANDLINE LOCK key CONFIG SET COMMANDLINE UNLOCK matching-key The commandline lock command will lock the commandline in the dialog and set the unlock key to match the 'key' parameter. This command is only valid when the command line it NOT locked. This command can be sent from any command source, not just from the command line. The commandline unlock command can also be sent from any source and not just the commandline itself. If the commandline is currently locked and the key on this command matches the key set with the previous commandline lock command, the commandline will be unlocked. When the commandline is locked, only GET, CONFIG GET, and CONFIG SET COMMANDLINE UNLOCK commands will be accepted from the command line. All other commands will return an error response of "Commandline locked". Also, the various buttons in the dialog, including the Close button are locked out until the interface is unlocked.

4) Made sure that the droplist with the ASIO device name is set in all cases when setting the interface from the command line.

5) Added a CLOSE <channel-list> command to stop playback on the list of channels and then close the files that were in use. Normally files are kept around in case they might be used again shortly. However that makes it difficult to
delete one of the files if you want to replace it with a new version of the file.

6) Fixed some bugs in buffer LRU list handling that could result in crashes in rare cases.

7) Fixed some bugs in keeping track of files when the same file is open on multiple channels.

8) It was possible to play a channel with no file loaded on it and not getan error. When you later loaded a file onto the channel it started playing immediately, which was generally unexpected. Play now checks that there is a file present on the channel before attempting to play and will send an error response if the channel isn't able to play.

9) Sped up the code that computes VU values.

10) Reduced signal generator overhead when not generating output (which is the usual case).

11) SMS will now accept temporary license files in place of a dongle. These files only live for a few days, but it is sufficient time to receive your actual dongle in the mail. A temporary license file is tied to one system and cannot
be moved between systems as a real dongle can.

12) When running on a temporary dongle file, this is a BIG ANNOYING MESSAGE that comes up to remind you that you are on borrowed time and it is running out. It will show the number of days you have remaining until the system reverts back into demo mode.

13) To get a temporary license file you must supply RSD with the machine id for the system you wish to license. You can get this number from the About Box

when the system is running in demo mode. The number is not available when you are running with a dongle or a temporary license file.

14) Fixed a potential deadlock that could happen during startup.

15) Fixed an annoying complaint that could occur during ASIO open if the dongle had an incorrect sample rate configured.

16) Fixed some problems with gain setting that made recent versions not work with the Waterworld show.

****

Version 1.0.23

****

1) If an output channel wasn't accumulating VU samples, we would erroneously keep repeating the last value VU sample rather than correctly returning 0.

2) Reimplemented the TCP message receive path to drastically reduce processor usage to something reasonable. Submaster changes in SMA no longer swamp the entire system with overhead.

****

Version 1.0.22

****

1) The license info now shows up in the About box, the splash screen, and is logged when the interface is opened.

2) If the interface is locked into a single operating mode by the licenseinfo
the interface cannot be switched to some other mode.

3) Changed the security dongle stuff to be able to decript the dongle contents on hopefully all MS systems, and not just most of them.

4) The Close button again works as God, Microsoft, Apple, and Linux expect a Close button to work -- it will Close the program. People that just want to MINIMIZE the window are directed to use the MINIMIZE BUTTON in the title bar, which has been designed specifically for this purpose, as stated in all of the system UI design manuals.

5) Fixed a problem where crosspoint gain was not set during a fade, but only at the end of the fade time (yuk).

6) Fixed another problem where fading a crosspoint up from 0 over time didn't work at all. Now it does.

****
Version 1.0.21
****

1) Implemented dongle code.

****

Version 1.0.20

****

1) Added an ugly hack to allow invalid channel numbers < 16 when talking to SMA. As long as at least one valid channel is given we will act on the command, ignoring the invalid channels.

****

Version 1.0.19

****

1) Implemented an assortment of channel status request commands. This is not an exhaustive set of possible status requests, but it covers many of the most common needs. A channel status request can request a common status type (such as gain or delay or mute status) for multiple channels of any type. The response for all channels will be returned on a single line.

All status response lines have a common format:

<status_type> <channel_id> = <status_value> ... ; <newline>

The status_type identifies the type of channel status on this line. All channels on the line are returning the same kind of status. The status type is always a single word with an initial capital letter and usually the remainder of the word in lower-case. Each channel, even if there is only one, is identified by the short form of the channel identifier. This consists of one or two letters and the appropriate channel numbers. The status value will be appropriate for the status being returned.

For numeric values this can be an integer or a floating-point number, as appropriate. For an on/off status such as the mute status of a channel it will be ON or OFF. For the channel name it will be the name string in all uppercase, or "None" (without the quotes) if the channel does not have a name assigned.

The following table shows the currently valid channel status commands, the response status_type value, and indicates the type of value to expect. All requests are of the form

GET CHANNEL <channel_list> <word>

Request	Response	Type	What
NAME	Name	string	channel name or 'None'
GAIN	Gain	floating	the channel gain between 0..N
GAINDB	GaindB	floating	the channel gain in dB
GAINMIDI	GainMidi	integer	gain between 0..127
PHASEREVERSE	PhaseReverse	ON | OFF	channel polarity status
MUTE	Mute	ON | OFF	mute status
SOLO	Solo	ON | OFF	solo status
DELAY	Delay	floating	the delay in seconds
TRACK FILE	TrackFile	"string"	the full path to the file
TRACK STATUS	TrackStatus	PLAY | STOP	playback status
TRACK POSITION	Position	integer	playback position in samples
TRACK TIME	TrackTime	floating	playback position in seconds
TRACK LENGTH	TrackLength	floating	track length in seconds
TRACK START TIME	TrackStart	floating	track start offset in seconds
TRACK STOP TIME	TrackStop	floating	track stop offset in seconds
TRACK REPEAT TIME	TrackRepeat	floating	loop-to point in seconds

2) The GET VU command is implemented. It returns average and peak VU readings for input and output channels, but not for crosspoints. The VU values are expressed as integers in the range of 0 to 255. These are suitable to be fed

into a VU meter with this range from minimum to maximum. The VU meter should be linear over this range; the values have been pre-warped to display reasonably.

The command has the format

GET VU <channel_list>

The response has the format

VU <channel_id>=<average>:<peak> ... ;

Avoid doing VU requests more often than every 100ms or so. VU processing is expensive, and the time is generally better used for other things.

3) Converted the Server to run in demo mode if there is no dongle. In demo mode the server runs in AudioBox-emulation mode with 2 input channels, 2 output channels, and 4 playback channels, at no more than 48K sample rate.

4) CONFIG SET INTERFACE now allows the sample rate to be specified. Previously a separate setup command was required.

CONFIG SET INTERFACE number | "name" INPUTS count OUTPUTS count PLAYBACKS count SAMPLERATE speed MODE NORMAL | AUDIOBOX | ABEMULATION | COMMANDCUE

5) Various CONFIG GET commands exist. This documents the commands and what they return. These commands do not require an open interface:

CONFIG GET VERSION Returns the current SMS version number string: Version 1.0.19.0 for example

CONFIG GET INTERFACES Returns a list of the available ASIO interfaces terminated by ".":

Interfaces 2 Interface 0 "ASIO 2.0 - Maya 7.1" Interface 1 "MOTU FireWire Audio"

CONFIG GET INTERFACE [n] Get information on the specified interface number:

InterfaceInfo 1 "MOTU FireWire Audio" Inputs 18 Outputs 18 DefaultSampleRate 48000

These commands require an open interface

CONFIG GET INTERFACE

When issued without an interface number this returns info on the current interface, including any limitations on the number of channels:

InterfaceInfo 1 "MOTU FireWire Audio" Inputs 2 Outputs 2 DefaultSampleRate 44100

CONFIG GET INPUTS

Returns the number of inputs available:

Inputs = 2

CONFIG GET OUTPUTS

Returns the number of outputs available:

Outputs = 2

CONFIG GET PLAYBACKS

Returns the number of playback channels available:

Playbacks = 4

CONFIG GET DEMO

Returns 1 if in demo mode, 0 otherwise:

Demo = 1

CONFIG GET DONGLE

Returns 1 if the dongle is detected, 0 otherwise:

Dongle = 0

CONFIG GET MODE

Returns the matrix operating mode:

Mode = AUDIOBOX

Possible values are NORMAL, AUDIOBOX, COMMANDCUE.

CONFIG GET SAMPLERATE

Gets the current interface sample rate:

SampleRate = 44100

6) Output channels now zero the VU reading when muted, as they should.

7) Input VU readings with SMS are post-fade rather than pre-fade as they are in a real AB. This has always been the case, this just documents that this is what is intended to happen.

8) SMS will now show demo mode and missing dongle information in the titlebar.

9) The log window will now expand wider when dropped down, making it easier to read long log lines. If SMS is too near the right edge of the window the dropdown will be trimmed at the right edge of the screen so that the scroll bar doesn't disappear.

****
Version 1.0.18

****

1) Changed a lot of strings to SoundMan-Server from SoundMan Server.

****

Version 1.0.17

****

1) Added code to insure that the ASIO callback thread is running at highest priority after lowering the GUI thread to normal priority. Some insane ASIO drivers seem to somehow use the main thread as the ASIO driver thread!

2) When the close button is clicked and the interface is open, just minimize the window instead of asking if you want to end the app.

****

Version 1.0.16

****

1) Extended SET MATRIX ROW to allow GAINMIDI and MIDI gain values in the range of 0..127. This simplifies some code in SoundMan-Assistant.

2) Extended SET MATRIX ROW to allow a FADETIME|FADETC parameter on the end to set the gain for all specified crosspoints at a non-zero rate.

3) Cleaned up the startup sequence so that we don't try to lay out the VU meters three different times.

4) Found that the VU meters weren't laid out correctly if the window was iconic when the meters are laid out. Added code to redo the layout when the window is finally displayed.

5) Slightly enlarged the size of the display for the connected ASIO device so that more of the text should be visible.

6) Added some experimental timecode decoding logic. This is not yet usable.

7) The ECHO driver for the GINA 24 sets the GUI thread to realtime priority. This results in audio breakup any time the interface is touched. Added code to change it back to normal priority like it should be.

****

Version 1.0.15

****

1) Playback channel soloing didn't work. It now does, and properly cross-mutes with the input channels.

2) SET CHAN P0-3 TRACK "NAME" will now correctly load the same file onto all of the selected playback channels. Previously only the first channel would have been loaded.

3) Fixed a timing window that in very rare conditions could cause a crash.

4) Under some conditions when starting many channels at once, some would not always start.

5) SET MATRIX ROW will now accept a row number, an input channel name, or a playback channel name to designate the crosspoint row to be set.

6) A crosspoint can be named, and referred to by name in commands that take a crosspoint identifier. The crosspoint can also be referred to by input_name.output_name, or by the usual number.number.

****

Version 1.0.14

****

1) "set chan i1 port none" did not work correctly.

2) Audio playback logic has been moved to detachable input nodes from the main playback channels. This allows any input channel to attach to a playback stream in Command Cue mode. It also makes it easier to properly implement multiple sample formats for wave files.

3) Channels can now be named. The syntax is SET CHANNEL <chan> NAME <name>. The name string cannot contain spaces or special characters other than an underscore and must start with an alphabetic character.

4) Port selection in Command Cue mode can now be done by name.

5) Channels in commands may now be specified by name as well as number. For an input or output channel the name can be given as

<type> <channel name> or just <channel name>

For a crosspoint or playback crosspoint channel the syntax can be

<type> <input name>.<output name> or <type> <crosspoint name>

Note that you can leave the channel type off for an input or output channel but you have to give it for a crosspoint or playback crosspoint channel.

Channel names do not have to be unique. However, if there is more than one channel of a given type with the same name, only the lower-numbered channel will ever be found by name. It is perfectly all right though to give both an input and an output channel the same name. The input channel will be found if no <type> is given, and either can be found if the correct <type> is given.

Example:

```
set chan in  1 name fred
set chan in  2 name wilma
set chan in  3 name barney
set chan out 1 name dsl
set chan out 2 name cluster
set chan out 3 name dsr
set chan wilma gaindb -2
set chan fred gaindb -5
set chan barney gaindb 0
set chan x wilma.dsl gaindb 0
set chan x fred.dsr gaindb 0
set chan x barney.cluster gaindb -2
set chan dsl-dsr gaindb 0
set chan cluster mute on
```

6) Added 'set <channel> track path "pathname"' command. This lets you set a path to all of the files that will be played on this particular channel so that you don't have to enter a full path on every file. You can still override the path by giving a full path as a file name. You can clear the path by setting it to an empty string.

7) The window position will now be remembered on application close and the position will be restored the next time the program runs.

8) Added a "CONFIG CLEAR LOG" command to clear the history buffer. This can be useful for applications that leave the server up for extended periods of time. For instance, a command could be sent to clear the log at midnight.

9) Added 128 group masters, named G0 to G127.

10) Added SET GROUP Gnn CHANNELS <channel list> to allow groups to master channels and other groups.

11) Added G | GROUP nn syntax to the channel list syntax so that you can mix normal channels and groups of channels in the same command.

****

Version 1.0.13

****

1) Again recognize "mode abemulation" to make the old SM-AB setup happy.

2) Fixed crosspoint GAINMIDI to set the live gain rather than the submaster gain if we aren't in CommandCue mode (and didn't ask for a log fade.)

3) Fixed a problem where we re-laid out the vu meters on every command from SM-AB!

4) The flag to do vu processing was not being managed correctly, so vu meter processing was random.

5) There was a race condition in shutdown that could cause crashes when deallocating the vu meters.

****
Version 1.0.12

****

1) Argh! The check for valid processor hardware got optimized out!

****
Version 1.0.11

****

1) Made the 30-day timeout actually work correctly.

2) Added VU meters on the output channels.

3) Various unrecorded simple bug fixes during runup to LDI 2006 in Las Vegas.

4) Changed the copyright to 2007.

****

Version 1.0.10

****

1) Fixed a largish handful of bugs all over the place. Most notably made sure that the floating point rounding mode is set correctly in all cases.

2) Now parse simple wave file headers correctly and decode the playback speed and number of tracks. Works for mono or stereo files with the standard sample rates.

3) Made 'config set samplerate' work correctly.

4) Added syntax to the 'set chan xxx track file "xxx" track n' command to let you pick which channel in the wave file will play back.

5) Added some basic Command Cue matrix support abilities.

   For hard input channels you can reconnect the input control cluster to another inputs in a mix and match form. Any control cluster can be assigned to any input and you can have more than one cluster on a single physical input. This corresponds to the typical Selector usage in Command Cue. Note this only works for hard inputs, not playback inputs or the signal generator!

   The crosspoint row has been split off of the input, and you can now assign an input cluster's output to any crosspoint row, or to no row at all. This corresponds to the typical Assignor usage in Command Cue. Again, you can have multiple input clusters assigned to a single crosspoint row, and crosspoint rows with no inputs.

   The default initialization is to map all physical inputs to an input control cluster and map that to the associated crosspoint row. If no changes are made in mapping the matrix will work normally.

6) Changed the 'config set mode' syntax to allow more input modes. The old syntax was 'config set mode abemulation [on|off]'. The new syntax is 'config set mode [normal|audiobox|commandcue]'.

7) Added 'set chan IN xx PORT n' to assign a particular hard input to this particular input cluster. The 'port xx' syntax is only valid on input channels, and only in CommandCue mode. Note that you cannot assign the input cluster to NO physical input as you can in Command Cue. You can get the same effect by muting the channel.

8) Added 'set chan [IN|PLAYBACK] xx ROW [n|NONE]' to select the crosspoint row to be used for this particular input or playback cluster. This is only valid for input and playback channels and only in Command Cue mode. By default each input cluster is mapped 1:1 to the same crosspoint row number.

9) Adjusted the log fade volume curve timing to match the old Command Cue hardware. Command Cue on a fade down from full by definition drops at 36db over the specified fade time. This means a complete fade to -144db will take 4 times the specified fade time. The time values for a CommandCue AutoPan fader should now match our time values and produce the same results.

10) Added a fade type of "Log" as a synonym for Exp or Exponential in the gain fade command format.

11) Set Matrix has been modified to set some or all of the crosspoints in a single matrix row to individual values for each point. The general syntax is: SET MATRIX ROW n [GAIN|GAINDB] channel [TO channel]@level... If neither GAIN nor GAINDB is specified the gains are in dB by default.

12) In CommandCue mode, a new command has been added to control the AutoPan fader separately from the main channel gain:

SET CHAN IN n AUTOPAN ON|OFF|UP|DOWN
LIN|LINEAR|LOG|EXP|EXPONENTIAL
FADETIME time
FADETC timecode

****
Version 1.0.9

****

1) Increased max playback channels to 1024.

2) Some assorted minor performance improvements.

****

Version 1.0.8

****

1) The server could crash in the socket handling stuff under conditions that I don't understand. Fixed the writes to be synchronous, which may help. Also added debug code to spit a message to DbgView if the socket is called from the wrong thread.

2) Changed eq band numbers to be 1-relative since I could never remember that they were zero relative before.

3) Changed the eq to allow positive gains up to +100db.

****

Version 1.0.7

****

1) Added commands to the parser to allow the signal generator to be configured from the interface.

2) Added the ability to sweep the signal generator over a frequency range in sine/square/triangle modes. This is a linear sweep, which is the most useful kind with an FFT analysis program as all signals will show the same level on a flat response.

3) Added the ability to do a 'pink sweep'. This will sweep a sine wave across a frequency range, but decreases the level at -3db/octave to match normal pink noise response.

4) Improved the pink noise generator. The ripple is now around .05db from a straight line rather than the 1.5db ripple previously.

****

Version 1.0.5

****

1) Fixed a bug in pitch and speed parsing.

2) Fixed a hang problem where a failed IO request would lock things up.

3) Fixed problems with playing off the end of a track, and with a track not looping when only the resume point has been set. Seemingly contrary to the command set documentation, the AB will loop from the end of the track to the resume point with only a resume point set.

****

Version 1.0.4

****

1) If the stop point was disabled the track wouldn't play. Fixed.

2) Changing the ASIO driver from SoundMan-AB might not load the driver when it should. Fixed.

3) Improved parsing of track start and stop times.

4) Now allow a submaster gain in dB for any channel or crosspoint. This was needed to make submasters and controllers work correctly in SoundMan-AB.

5) Channel and crosspoint gain is not allowed to go above unity in AudioBox emulation mode.

****

Version 1.0.3

****

1) Slightly reduced the processing overhead for several of the live channel input routines.

2) Added yet more debugging log messages to the process of opening an ASIO device.

3) Changed the parameters to the ASIOInit call to only put a window handle in the sysRef field if the driver name starts with "M-Audio", otherwise leave the sysRef field zero. This is documented as zero on Mac platforms, and a valid window handle on Windows platforms. NULL is a valid window pseudo-handle (meaning 'the desktop'), but the next-to-latest version of the M-Audio drivers will crash on exit if the handle isn't non-zero.

   On the other hand, it turns out Digigram ASIO drivers seem to think this field is some sort of an undocumented device index, and the driver will crash if it isn't zero!

4) Previously if we couldn't set the interface sample rate to 48000 we would give up. Now we check the interface sample rate after attempting to set it. If the rate is something other than 48000 we pop up a warning message but allow the use of the interface anyway.

   A possible reason the interface sample rate might not be settable is if the interface is working off of an external clock source. Depending on what the interface reports to us as a sample rate this might work out OK, or might produce totally bogus sound. The user can decide for himself.

5) After just an amazing amount of work, fixed a problem where loading a new track that was the same as the current track would not start playing from the front of the file as it should, but would just continue to play without stopping.

6) Previously playing a short track and then a long track on the same channel would end up stopping at the time the first track ended. This is fixed, and the longer track will now correctly play to the end or stop point.

****

Version 1.0.2

****

1) No longer crash with a divide by zero calculating processor utilization.

2) SoundMan-AB no longer opens ASIO devices, possibly contending with the server.

3) Removed some probably unneeded code that was triggering a bug in the ECHO Layla/Gina24 drivers. The drivers now seem to come up and work fine, at least on my system.

4) Added logging of AsioResyncRequest, which probably indicates data loss occurred. Not all drivers will make this message, but the Echo drivers do.

5) Added logging code to log all possible bailouts from sound device initialization. This will make it easier to see why some interfaces don't work.

6) Fixed a problem with crosspoint linking and delinking that could end up getting the links broken and locking up the system.

7) Added an option to the interface to set the server to "below normal" priority instead of the usual Realtime priority. At low priority the audio can be expected to click and burp as you do other things. However, if the server gets itself into an internal loop (as happened sometimes in version 1.0.1 when changing crosspoint gains) the system won't get locked up and you will be able to kill the server.

8) Added a button to copy the command history log to the clipboard. From there you can copy it into a Notepad or mail message window. The command history could help me solve problems when they occur.

9) Selecting "None" for the ASIO interface won't spuriously try to open a driver with that name.

****
Version 1.0.1

****

1) Initial release of SoundMan-Server.