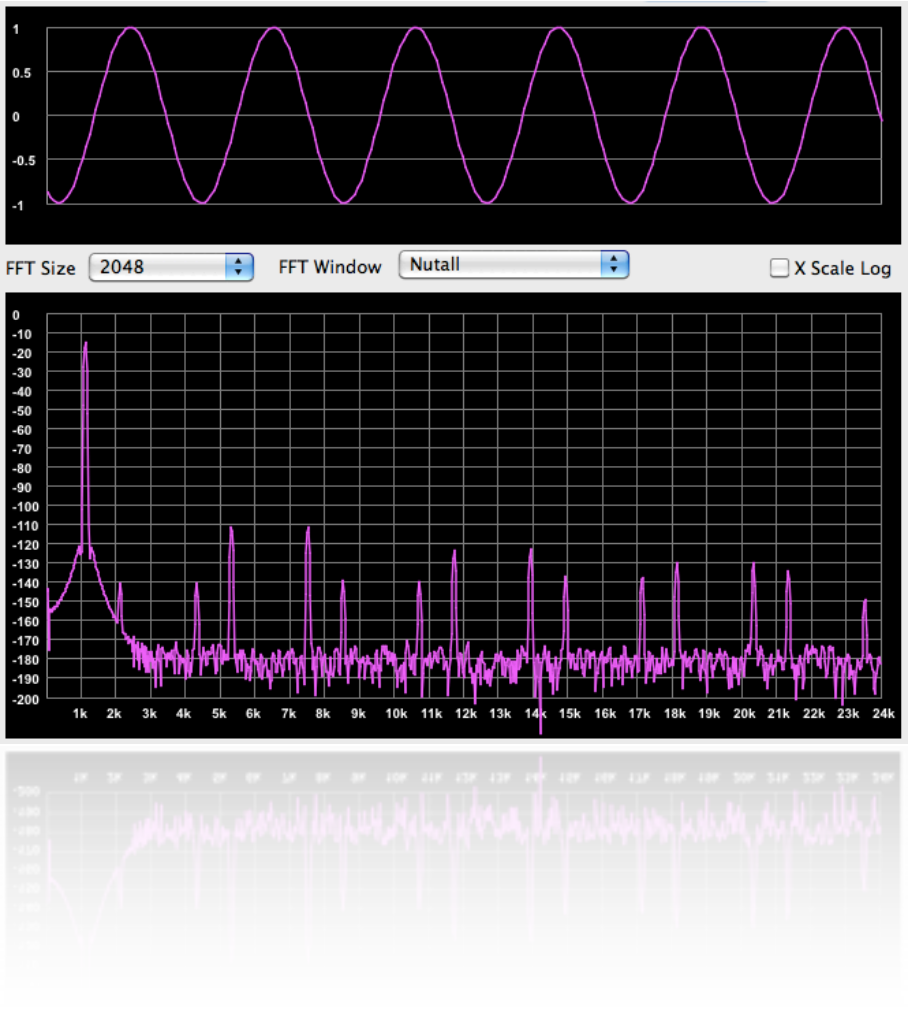
AN-004
APPLICATION NOTE

SLIMbus
Data Channel Content Generation



LnK
44, rue des Combattants
B-4624 Romsée
Belgium
www.lnk-tools.com
info@lnk-tools.com

Table of Content

1. Purpose of the document

This document will describe in detail all the steps that are necessary to achieve a distortion (or error) free data streaming using the Traffic Generator.

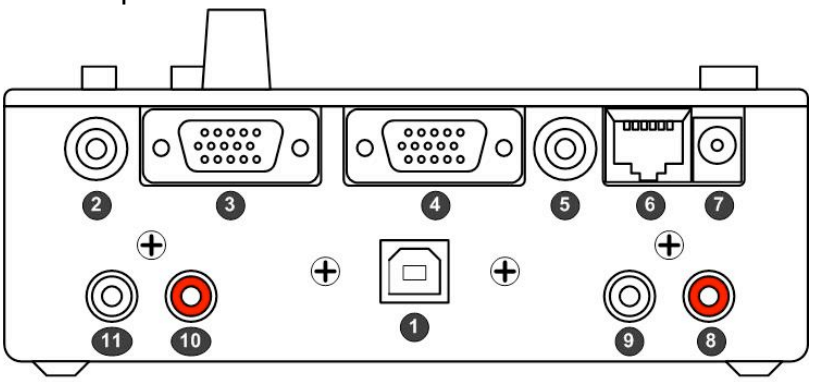
We will use the **LnK** SLIMbus Audio Bridge as a SLIMbus audio device and the Audio Precision APx585 audio analyzer for the purpose of the exercise.

A basic knowledge of the SLIMbus specification and of the operation of the SLIMbus Audio Bridge is required to follow the various explanations given in this document. Refer to the SLIMbus Specification document and to the SLIMbus Audio Bridge HW and SW user manuals for more information.

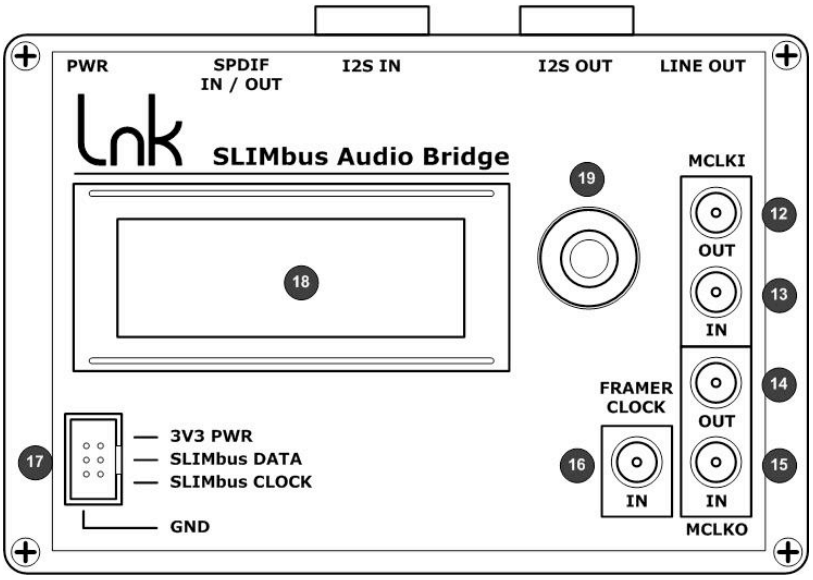
2. Presentation of the equipments

2.1. SLIMbus Audio Bridge

The tests described in this document will require the SLIMbus Audio Bridge. An audio analyzer will also be required for the audio tests. Any kind of audio analyzer is suitable. However, the APx500 models from AudioPrecision are especially suited for these tests. For detailed information about these tools, please refer to their respective user manual.



1 - USB connector
2 - Analog line output (mini-jack)
3 - I2S multichannel output
4 - I2S multichannel input
5 - SPDIF IN / OUT (mini-jack)
6 - Factory Service
7 - Power supply
8 - SPDIF OUT
9 - SPDIF IN
10 - Analogue OUT (R)
11 - Analogue OUT (L)

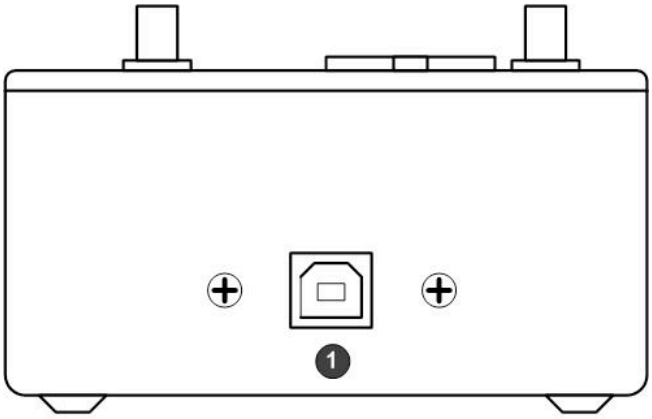


12 - Audio Master IN clock out
13 - Audio Master IN clock in
14 - Audio Master OUT clock out
15 - Audio Master OUT clock in
16 - Framer clock input
17 - SLIMbus connector
18 - Display
19 - Rotary knob + push

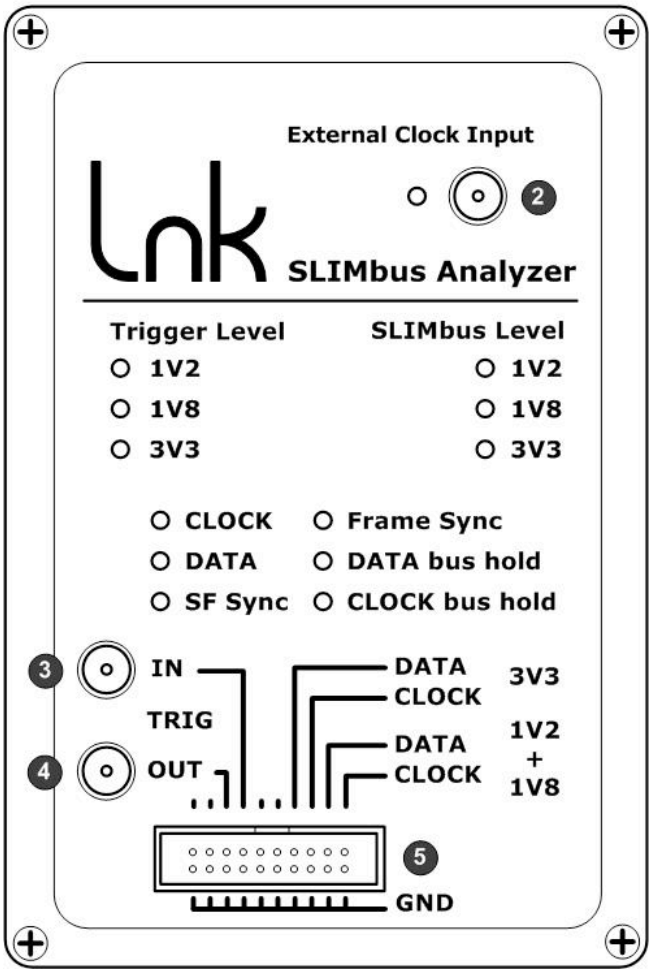
The SLIMbus Audio Bridge will transmit (or receive) digital audio streams on (from) SLIMbus. The bridge features SPDIF input and output and I2S multichannel input and output. The I2S interface use the Left Justified data format and 64 bits per frame. The I2S IN/OUT multichannel interface is pin/signal compatible with the AudioPrecision DSIO interface of the APx500 machines when using the special cables delivered with the bridge.

The 8 port bridge needs a power supply and needs to be connected to a PC through USB to be able to feed the SLIMbus messages on the bus.

2.2. SLIMbus Protocol Analyzer



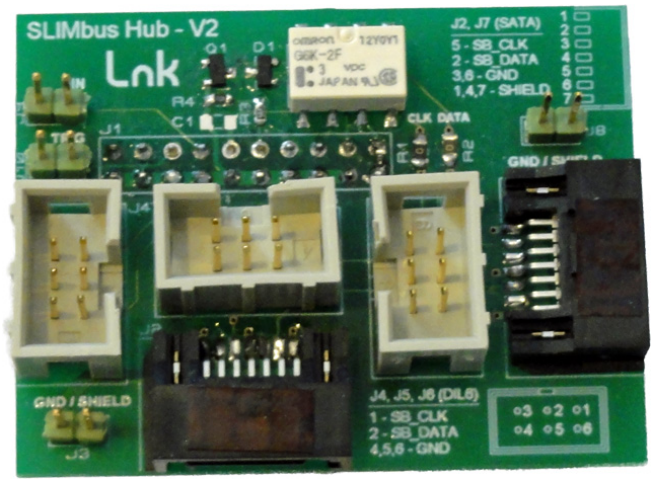
1 - USB connector



2 - Framer Clock Input
3 - Trigger input connector
4 - Trigger Output connector
5 - SLIMbus connector

The Analyzer / Generator hardware needs to be connected to a PC through USB. The SLIMbus Protocol Analyzer software must be launched to operate the analyzer hardware. The Protocol Analyzer will capture and decode all the SLIMbus traffic. The Traffic Generator will build and transmit the SLIMbus stream, especially the messages needed to configure the bus and the Audio Bridge to setup audio channels and transmission.

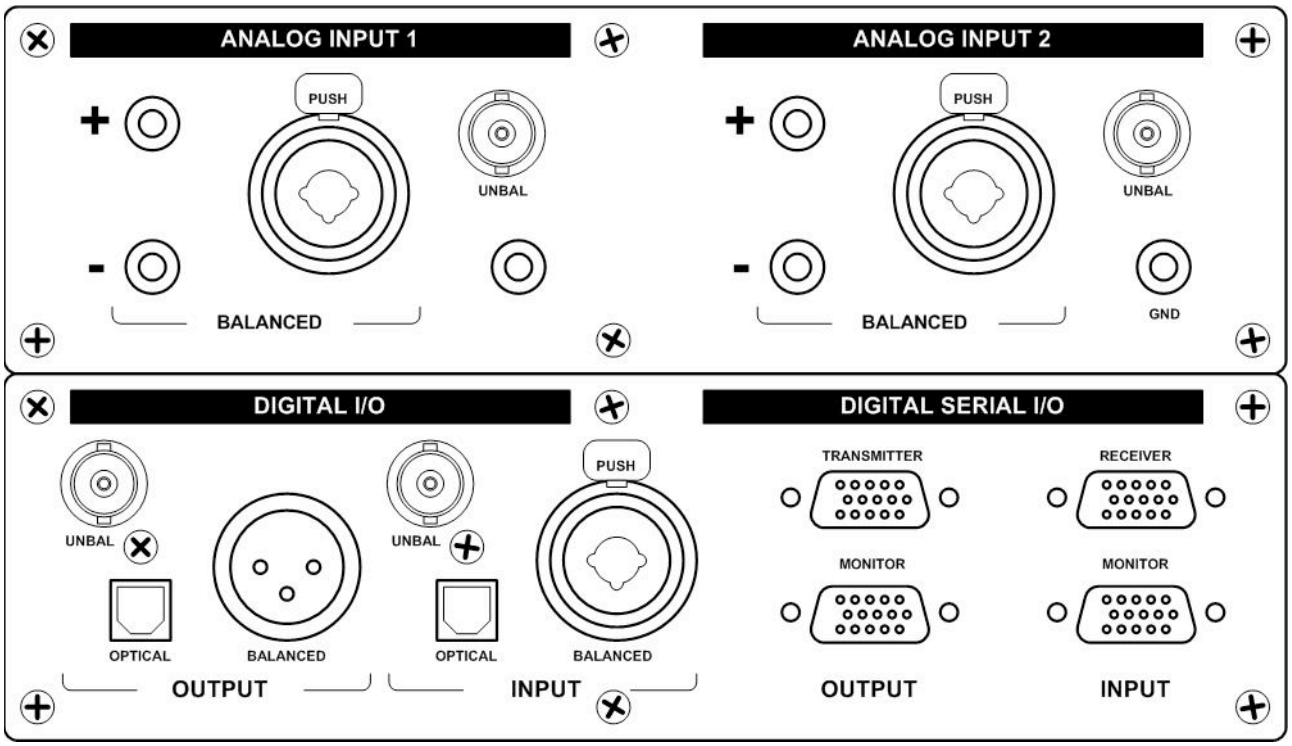
2.3. SLIMbus mini hub



This little board shall be plugged on the SLIMbus Protocol Analyzer hardware, in its SLIMbus connector. It provides SLIMbus signals on three DIL06 connectors and on 2 SATA connectors. The use of the mini hub is not mandatory but makes the connection to the SLIMbus Audio Bridge pretty easy.

2.4. Audio Analyzer

The audio Analyzer shall provide at a minimum a SPDIF input, a SPDIF output on coaxial connector (unbalanced) and 2 unbalanced analog inputs. For the multichannel tests, a DSIO (I2S multichannel) interface is needed.

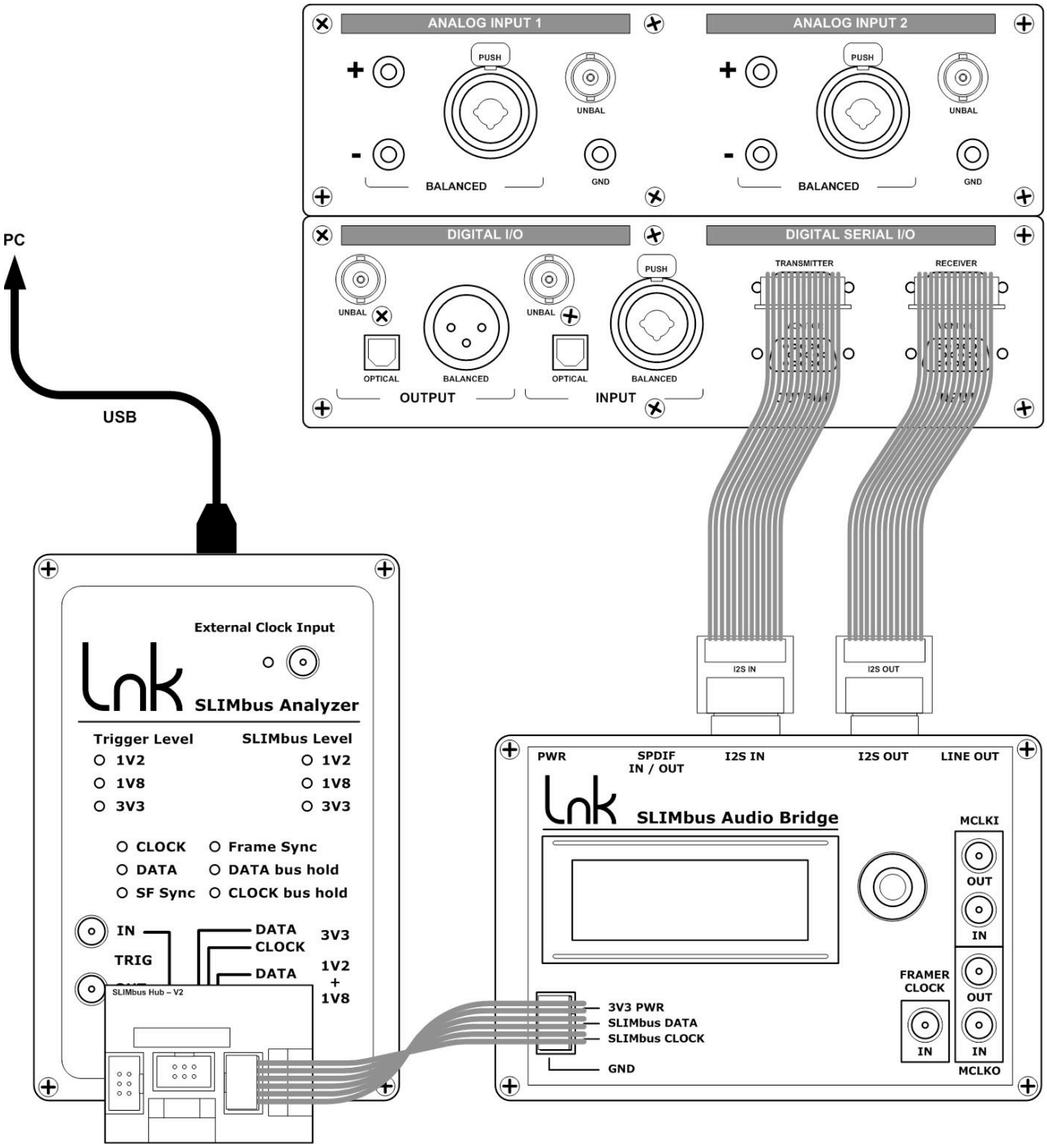


3. Sine wave generation

In this section, we will focus on data generated by the Traffic Generator itself. We will use the sine waves but any other waveforms available in the tool is usable with the same procedure.

3.1. Hardware setup

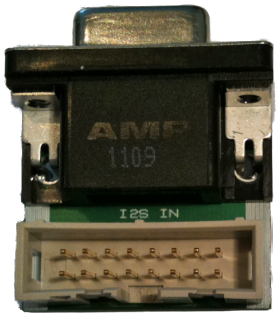
The Audio Bridge and the Audio Analyzer are used only to show FFT figures of the generated content (as independent mean of signal quality assessment). The main unit of interest here is the SLIMbus Protocol Analyzer / Traffic Generator.



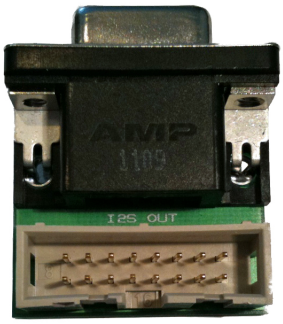
A pair of I2S cables and adapters are delivered with the Audio Bridge to ease connection to a variety of audio analyzers. However, when it is about testing I2S multichannel, the best option is to use an Audio Precision APx585 or APx525 model with the DSIO option.

There is a small difference between the I2S IN and I2S OUT cables. Do not exchange them. The streaming will not work anymore (swap of the bit clock and word clock).

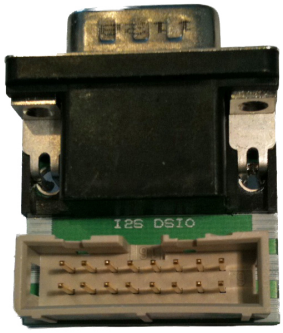
I2S cable assembly



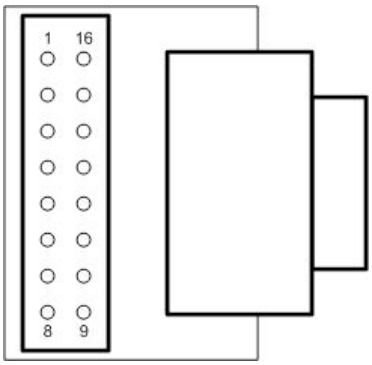
I2S IN adapter



I2S OUT adapter



I2S DSIO adapter



The I2S IN & OUT adapter pin assignment is as follow:

1 - BCLKI (Bit clock at 64Fs)
2 - MCLKI (Master clock at 128Fs or 256Fs)
3 - GND
4 - SDATA1 (Serial data stream 1)
5 - SDATA2 (Serial data stream 2)
6 - SDATA3 (Serial data stream 3)
7 - SDATA4 (Serial data stream 4)
8 - LRCLKI (Word or Sync clock at 1Fs)
9..16 - GND

3.2. Isochronous streaming

We will start with the simplest case. In an Isochronous stream, the SLIMbus channel rate and the sample rate (or Presence Rate in the SLIMbus terminology) are rigorously equal. The test script will create one audio stream and feed it with a sine wave.

Before we start creating the test script, a few words need to be said about the Traffic Generator HW memory limitations. The size of the HW SRAM limits the number of different superframes to 1100. With a bus running at 12.288 MHz, the superframe duration is equal to 6144 bits / 12288000 Hz = 500us. The 1100 superframes will therefore last 0.55s. If this duration is sufficient to the user, there is not much more to say about the topic.

However, if a long lasting stream is required, care must be given to the script creation to avoid signals artifacts.

We will use the loop capability offered by the hardware. A loop will repeat the superframes it contains a given number of time. Loops can run indefinitely (when the loop Repeat parameter is greater than 32768), if required, providing a never-ending stream.

However, because loops are repeating at superframe level, there are many constraints we need to look at.

a) The Whitening Signal (a bit with a pseudo-random pattern in the Framing Information bits) will not be compliant to the specification anymore. In practice, this is not an issue.
b) The Phasing signal that operates by bulk of 10 superframes will also get corrupted. In practice, it should also not be an issue.
c) The data channel wave forms generated by the Traffic Generator must have a period that relates to the superframe duration. We will have a deeper look at that.

Let's start with a tone at 1 kHz and a sampling rate of 48 kHz.
Such a tone will get 48000 / 1000 = 48 samples per period.

If we take a SLIMbus Root Frequency equal to 24.576 MHz (RF=1) and a Clock Gear equal to 9 (CG=9), the actual bus frequency will be equal to 12.288 MHz.
A superframe always contains 6144 bits.
The superframe duration is therefore equal to 6144 / 12288000 = 500 us.
Such a superframe will carry 24 samples of the 1 kHz tone. We need 2 superframes to convey a complete period of the sine wave.

The script to generate the sine wave is pretty simple. It consists in creating one data channel. We added a message to connect that data channel to one port of the Audio Bridge.

▸ 0		Boot Sequence	3072
▸ 1		Superframe Begin	2, SM=19, CG=9, RF=1, 12, Auto
▸ 2		Loop Begin	10
▸ 3		Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto
▸ 4		Loop End	
▸ 5		Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto
▸ 6		ASSIGN_LOGICAL_ADDRESS	Dst=0x01C100010200, LA=2
▸ 7		CONNECT_SINK	Src=0xFF, Dst=0x02, CN=0, PN=0
8		BEGIN_RECONFIGURATION	
▸ 9		- NEXT_DEFINE_CHANNEL	CN=0, SD=3140, TP=0, SL=6
▸ 10		- NEXT_DEFINE_CONTENT	CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=6
▸ 11		- NEXT_ACTIVATE_CHANNEL	CN=0
12		RECONFIGURE_NOW	
▸ 13		Loop Begin	65000
▸ 14	R	Superframe Begin	2, SM=19, CG=9, RF=1, 12, Auto
▸ 15		Loop End	

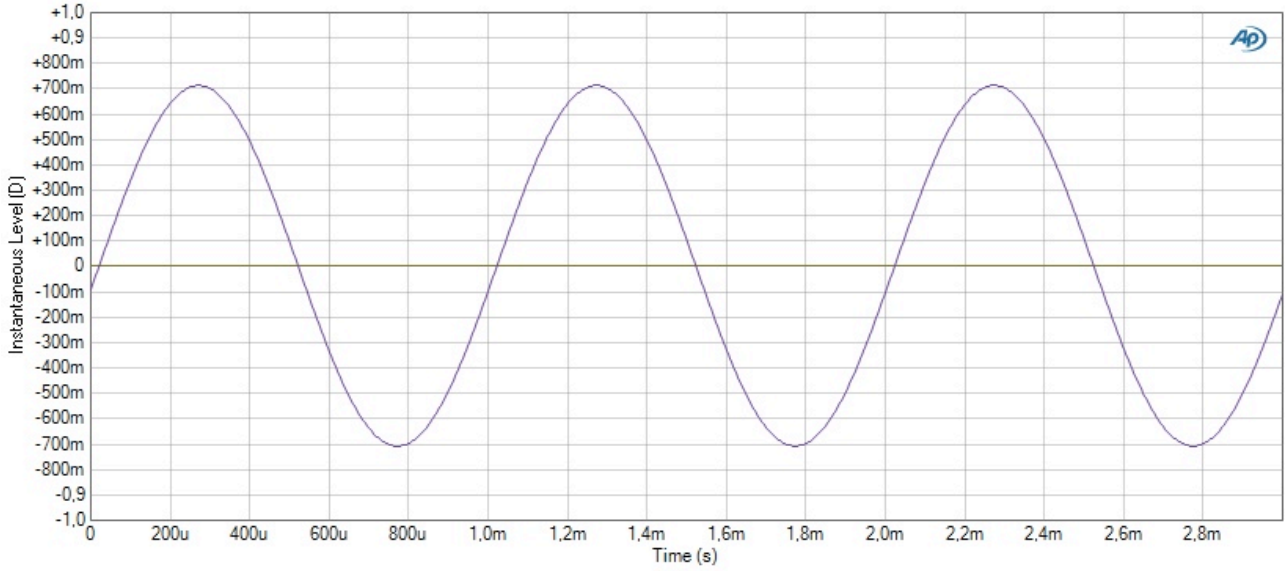
The data channel has a segment of 24 bits long and uses the Isochronous protocol. The channel rate and Presence rate (e.g. sample rate) are equal to 48 kHz. The last loop repeats endlessly the content of 2 superframes.

To add content to the data channel, we need to define a channel feed in the Traffic Initialization tab. Clicking on the "Channel Feed" button adds an entry in the list. We can now edit it to be sure that the channel number corresponds to what we have defined in the messages of the Reconfiguration Sequence.
The most important parameter is the "Signal Feed". It is set by default to "**Sine: 1kHz; -3dBFs**". We will keep this value as it serves the purpose of the test.

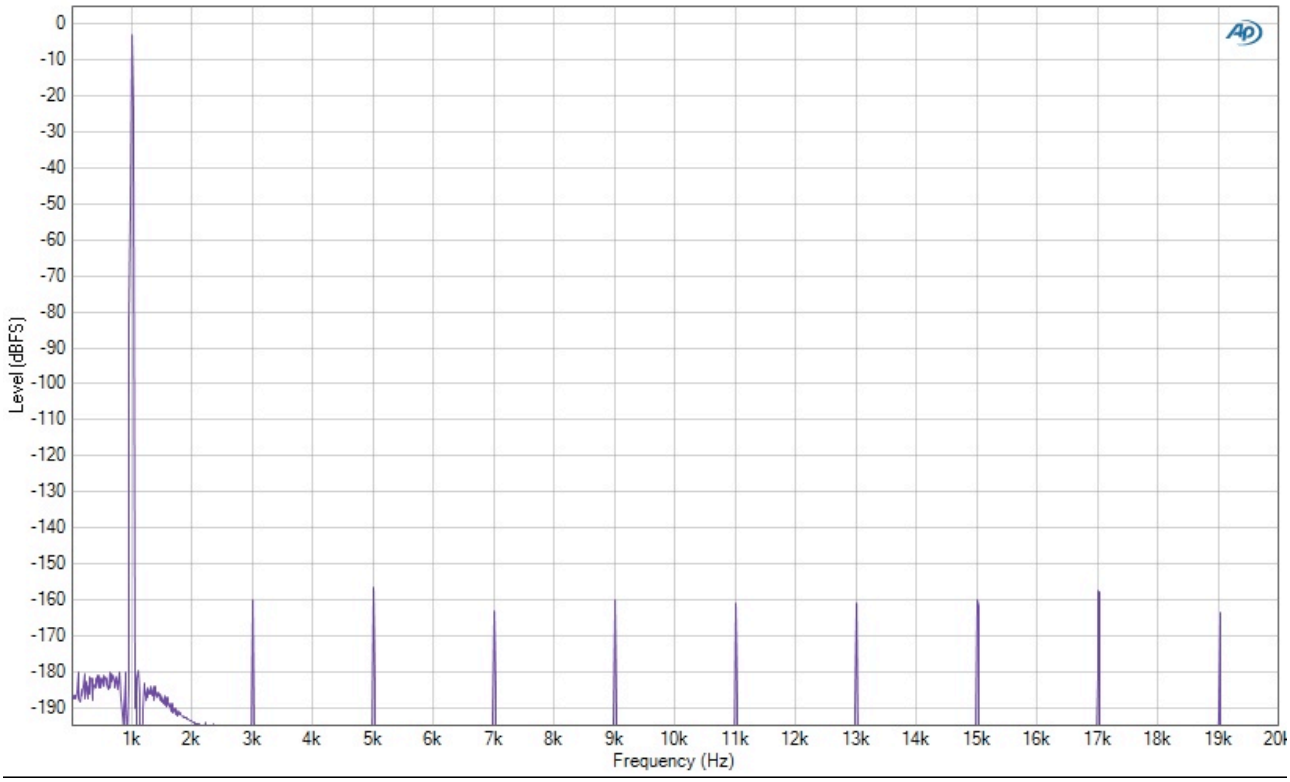
▸ 0	**HW configuration**	
▾ 1	**Framing Information**	
	Subframe Mode	19 – 4 / 32
	Clock Gear	9 – 6.4 to 14.4 MHz
	Root Frequency	1 – 24.576 MHz
▸ 2	**HW Guide Byte**	
▸ 3	**Framer**	
▾ 4	**Automatic MSG response**	
	Device Address (EA or LA)	0xFF
	Message Type	0xFF – ALL Types
	Message Code	0xFF – ALL MESSAGES
	Response	NORE
	Counter	0
▾ 5	**Channel Definition**	
	Channel Number	0
	Signal Feed	Sine: 1kHz; -3dBFs
	Segment Distribution	3140
	Segment Length	6
	Transport Protocol	0 – Isochronous Protocol
	Frequency Locked	1 – Frequency locked
	Presence Rate	3 – 48kHz
	AUX Format	0 – Not applicable
	Data Type	1 – LPCM audio
	Channel Link	0 – Channel not linked
	DataLength	6
	Sink	

The other parameters of the channel definition can be left blank. ScriptBuilder will automatically feed them when compiling the XML code.

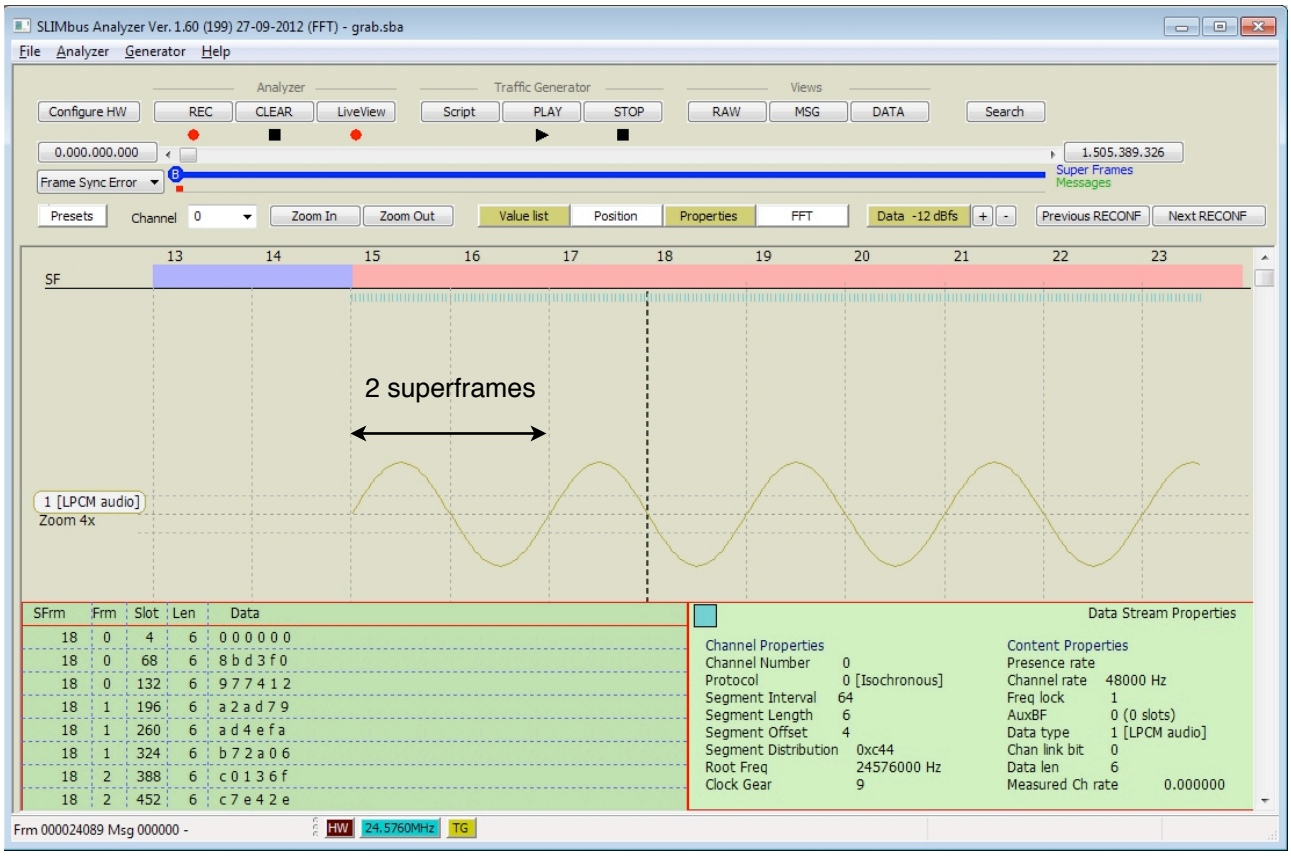
Let's play the scrip and capture the digital stream generated by the Audio Bridge with the APx585 audio analyzer.



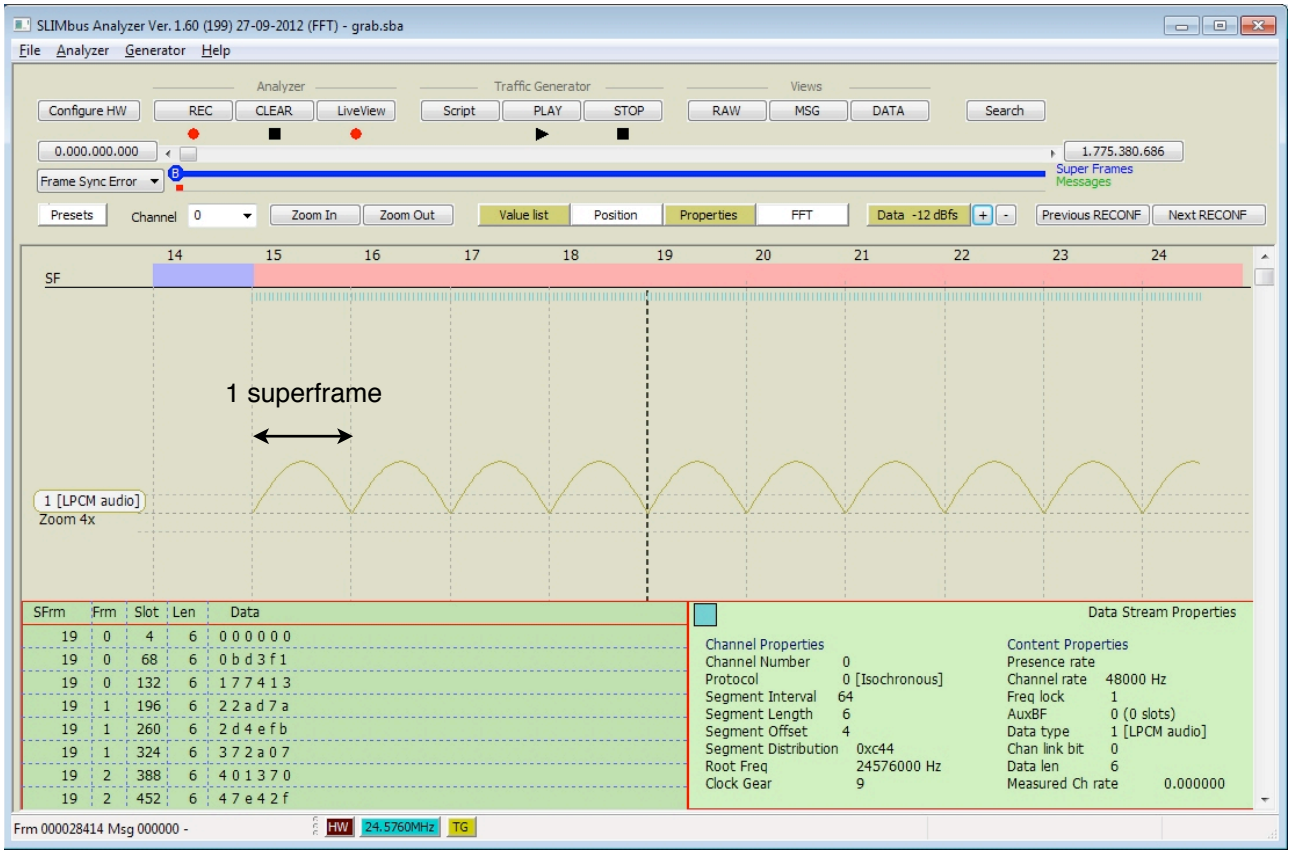
We can see that the captured sine wave looks pretty clean. However, a FFT will tell us more about the spectral purity of the tone. We can see that the harmonics are all below -150 dB, clearly showing that the residual THD is solely due to the lack of dithering on the LSB of the sample.



This very simple script is giving us a clean sine wave in an endless stream. Now, let's have a look to the Protocol Analyzer capture to better understand what happened. The audio stream effectively starts at superframe 15. We can see the sine period starting at the beginning of superframe 15 and ending at the end of superframe 16. The Traffic Generator has computed these 48 samples and the HW loop is repeating them endlessly.



If we only had 1 superframe in the repeat loop, we would have seen the first half of the period repeated endlessly. Only half of the period is contained in one superframe.



The number of superframes to put in the loop is really crucial to get the cleanest possible tone generation. As we saw, there is clearly a relation between the superframe duration and the sine period duration.

An integer number of signal periods must fit exactly in an integer number of superframes

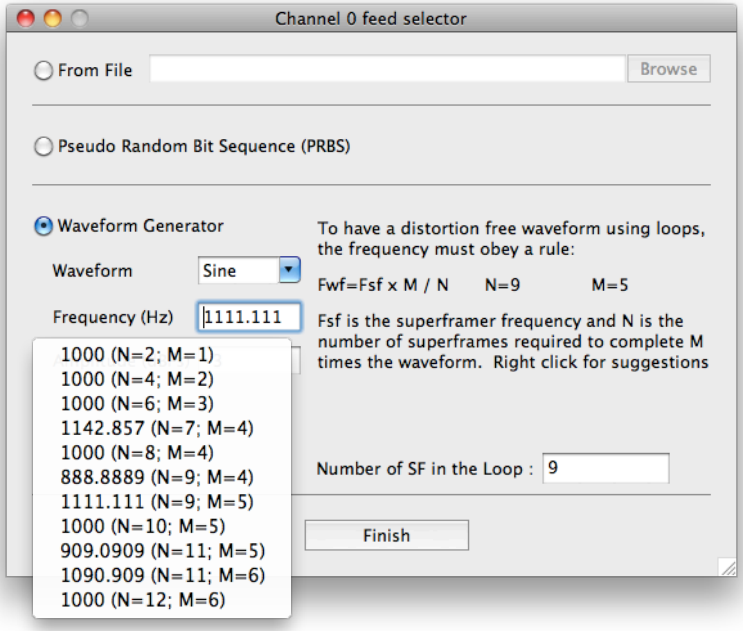
ScriptBuilder offers a tool to simplify the selection of a tone frequency that will be “loop” friendly.

Right-click in the “Signal Feed” parameter cell. A contextual pop-up menu will appear. Clock on “Source Selector”. A new window will appear. It allows to build the “Signal Feed” parameter value without making syntax errors.

Type the desired value for the frequency in the “Frequency (Hz)” text field. Then right-click to obtain a list of possible frequencies that will meet the requirement listed above. The list will contains the frequencies close to the desired frequency. Pick one in the list.

The number of periods (M) and the number of superframes (N) in wich the M periods a re fitting are listed. Once selected, the window will show the number of superframes to put in the loop.

In the example, the target frequency was 1000 Hz. We choose 1111.11111 Hz that will repeat 5 times over 9 superframes.



Selecting the right frequency through this window will also improve the accuracy of the generator. Instead of using the given frequency that is inevitably truncated, the Traffic Generator will use the ratio M/N to get a much more accurate value of the frequency.

3.3. Pushed streaming

Let's add a bit of complexity to the exercise. We will now generate a sine wave at 1 kHz with a sample rate of 44.1 kHz and a channel rate of 48 kHz. We will use the Pushed protocol to transport the 44.1 kHz sampling rate.

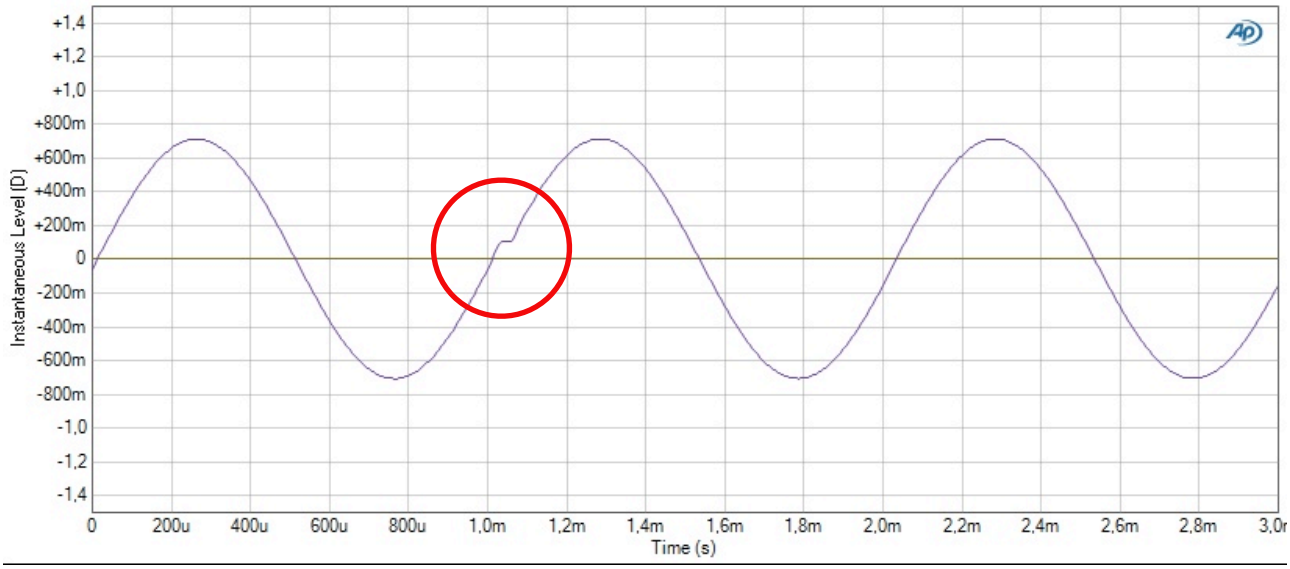
The segments will have an additional slot (4 bits) to carry flow information. In particular the Presence bit that indicates when a segment contains (or not) a valid audio sample.

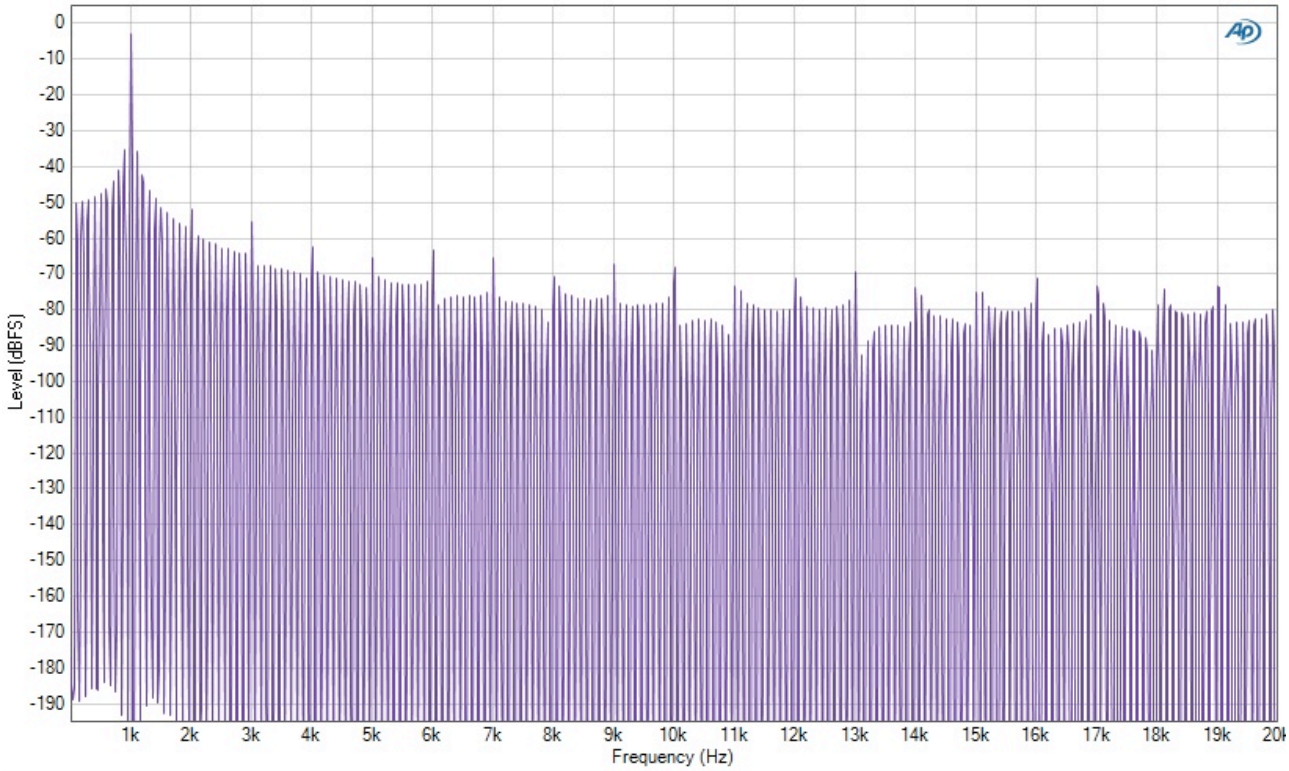
The channel rate being the same as before (48 kHz), the sine wave period will extend exactly over 2 superframes.

▸ 0		**Boot Sequence**	3072
▸ 1		**Superframe Begin**	2, SM=19, CG=9, RF=1, 12, Auto
▸ 2		**Loop Begin**	10
▸ 3		**Superframe Begin**	1, SM=19, CG=9, RF=1, 12, Auto
▸ 4		**Loop End**	
▸ 5		**Superframe Begin**	1, SM=19, CG=9, RF=1, 12, Auto
▸ 6		**ASSIGN_LOGICAL_ADDRESS**	Dst=0x01C100010200, LA=2
▸ 7		**CONNECT_SINK**	Src=0xFF, Dst=0x02, CN=0, PN=0
8		**BEGIN_RECONFIGURATION**	
▾ 9		**- NEXT_DEFINE_CHANNEL**	CN=0, SD=3140, TP=1, SL=7
		Channel Number (CN)	0
		Segment Distribution (SD)	3140
		Transport Protocol (TP)	1 - Pushed Protocol
		Segment Length (SL)	7
▸ 10		**- NEXT_DEFINE_CONTENT**	CN=0, FL=1, PR=11, AF=0, DT=1, CL=0, DL=6
▸ 11		**- NEXT_ACTIVATE_CHANNEL**	CN=0
12		**RECONFIGURE_NOW**	
▸ 13		**Loop Begin**	65000
▸ 14	R	**Superframe Begin**	2, SM=19, CG=9, RF=1, 12, Auto
▸ 15		**Loop End**	

The main difference is the Transport protocol that is set to Pushed and the segment size that is now equal to 7 slots (24 bits sample + 4 bits flow information). The Presence Rate in the NEXT_DEFINE_CONTENT is now equal to 44.1 kHz (PR=11).

If we play the script, we will see the following wave form and FFT spectrum.





Not as good as we could have thought... We see from time to time glitches in the wave form and the FFT shows a lot of spectral noise. The Traffic Generator is still repeating the 2 superframes endlessly.

How can it be possible that the glitch appears from time to time and not on every period?

To understand what's happening, we need to dig into the details of the transport of the Pushed protocol.

The data channel will have 48 segments during 1 ms (duration of 2 superframes) while there will be 44.1 samples. Here comes the problem. We cannot have 44.1 samples. It must be an integer number of samples.

If we look at the generated channel content, we get the following:

Sample 1	0X000000	**Sample 13**	0X5A9DD1	**Sample 25**	0X00A542	**Sample 37**	0XDA9CA3
Sample 2	0X0CDDFB	**Sample 14**	0X59BE7F	**Sample 26**	0X8C3A4F	**Sample 38**	0XD9D4CB
Sample 3	**NO SAMPLE**	**Sample 15**	**NO SAMPLE**	**Sample 27**	0X98DA72	**Sample 39**	0XD73AEB
Sample 4	0X197935	**Sample 16**	0X570D99	**Sample 28**	**NO SAMPLE**	**Sample 40**	**NO SAMPLE**
Sample 5	0X259048	**Sample 17**	0X529915	**Sample 29**	0XA4F9A5	**Sample 41**	0XD2DC82
Sample 6	0X30E47B	**Sample 18**	0X4C780E	**Sample 30**	0XB05905	**Sample 42**	0XCCD039
Sample 7	0X3B3B08	**Sample 19**	0X44CA52	**Sample 31**	0XBABD93	**Sample 43**	0XC53571
Sample 8	0X445E4E	**Sample 20**	0X3BB7B7	**Sample 32**	0XC3F165	**Sample 44**	0XBC339E
Sample 9	0X4C1EE4	**Sample 21**	0X316F4D	**Sample 33**	0XCBC4BC	**Sample 45**	0XB1F979
Sample 10	0X525494	**Sample 22**	0X26266D	**Sample 34**	0XD20F00	**Sample 46**	0XA6BC12
Sample 11	0X56DF25	**Sample 23**	0X1A17A2	**Sample 35**	0XD6AF8E	**Sample 47**	0X9AB5B7
Sample 12	0X59A708	**Sample 24**	0X0D817A	**Sample 36**	0XD98E66	**Sample 48**	0X8E24CC

We see that 4 segments are empty. So we get 44 samples over 48 segments. In practice, this leads to a 44.00 kHz sampling rate instead of the expected 44.1 kHz sampling rate. However, the Audio Bridge was told that the Presence Rate was 44.1 kHz so the generated sampling clock was effectively 44.1 KHz.

The Audio Bridge received from SLIMbus samples at a rate of 44 kHz and transmitted them to the Audio Analyzer at a rate of 44.1 kHz. 44.1/44= 1,0022727. A sample will be missing every 1 / 0,0022727 = 440 samples, so about every 10 periods (480 segments or 20 superframes). The bridge port FIFO is then getting an under-run event. A sample is therefore repeated twice, leading to glitch we see in the captured wave form.

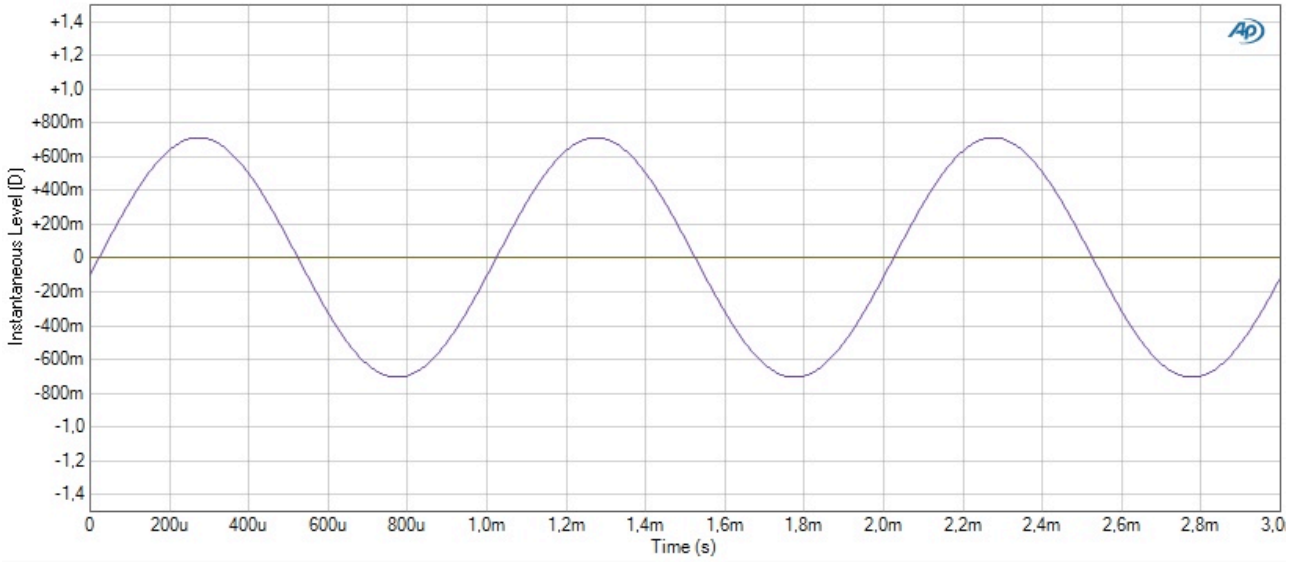
Now that we understand the mechanisms behind the Pushed protocol, let's see how we can improve the signal quality.

To get an integer number of samples over an integer number of segment, we need to find the rational ratio of 48000 / 44100. The result is 160 / 147. That means that 147 samples will be carried over 160 segments. 160 is called the Segment Repeat Period.

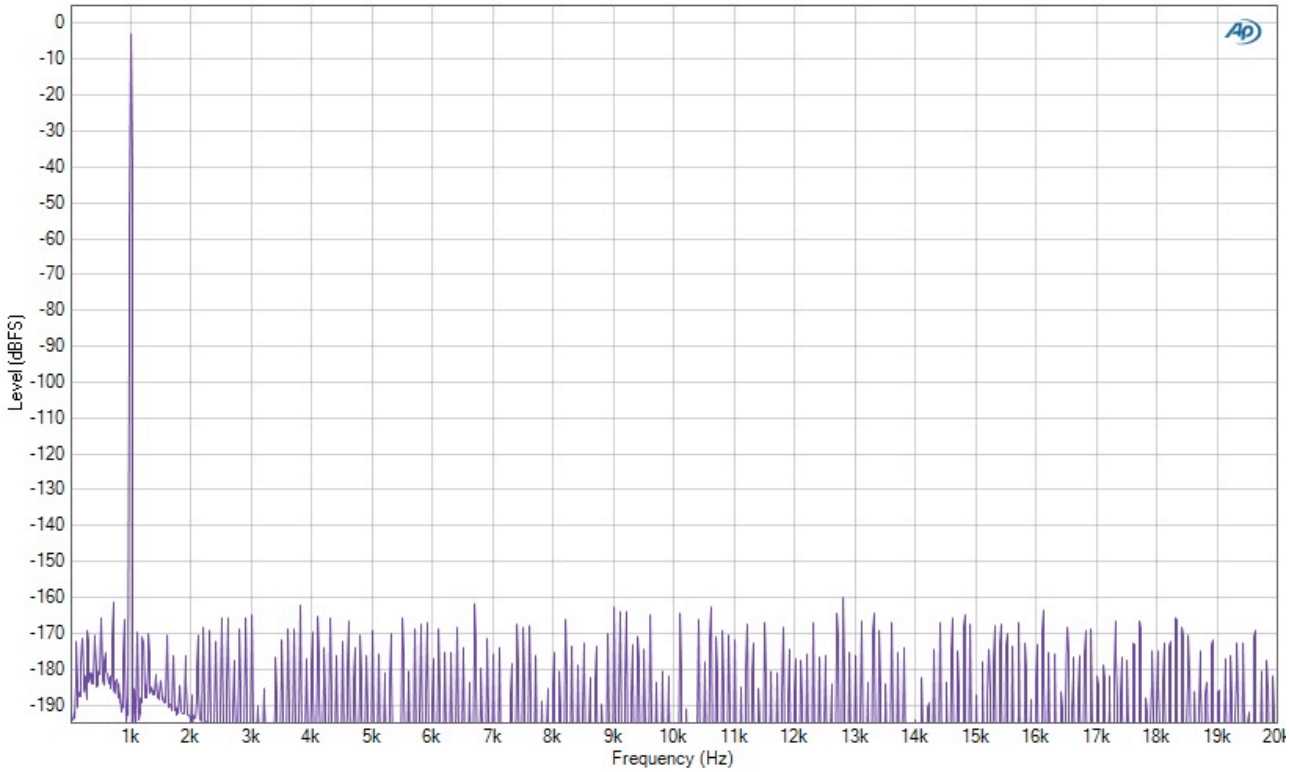
160 segments do not fit in an integer number of superframes. We must get a multiple of 48. The smallest possible number of segments is therefore 480, leading to 441 samples. We have an integer number of samples and an integer number of superframes (20). All the conditions are fulfilled to have a clean tone.

Let's replace the superframe repeat value 2 by 20 and run a measurement again.

The wave form looks cleaner but some glitches could still be hidden.



So Let's have a look to the FFT plot.



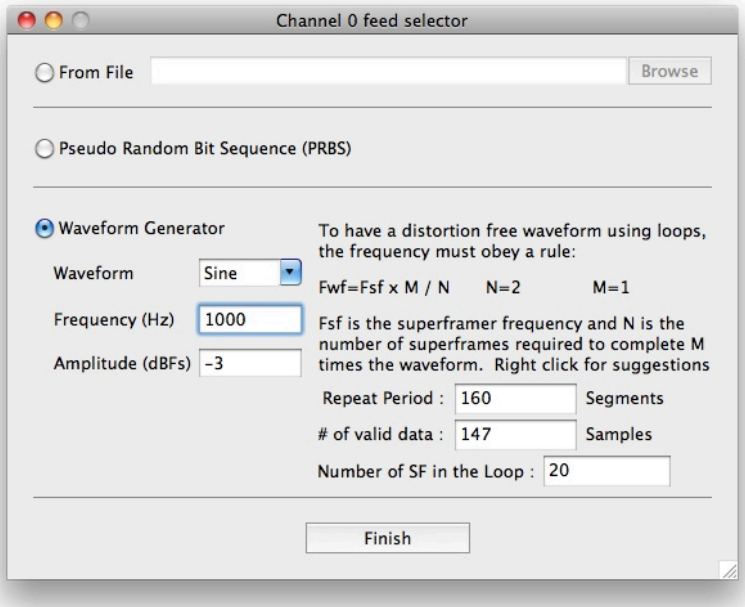
We can see that the overall THD+N is less than -144 dB, which is about the best that can be achieved by 24 bits wide samples. The streaming is clean.
By going from a loop repeat period of 2 superframes to a loop repeat period of 20 superframes, we went from a distorted sine wave to an ultra-clean sine wave.

An integer number of Segment Repeat Periods must fit exactly in an integer number of superframes

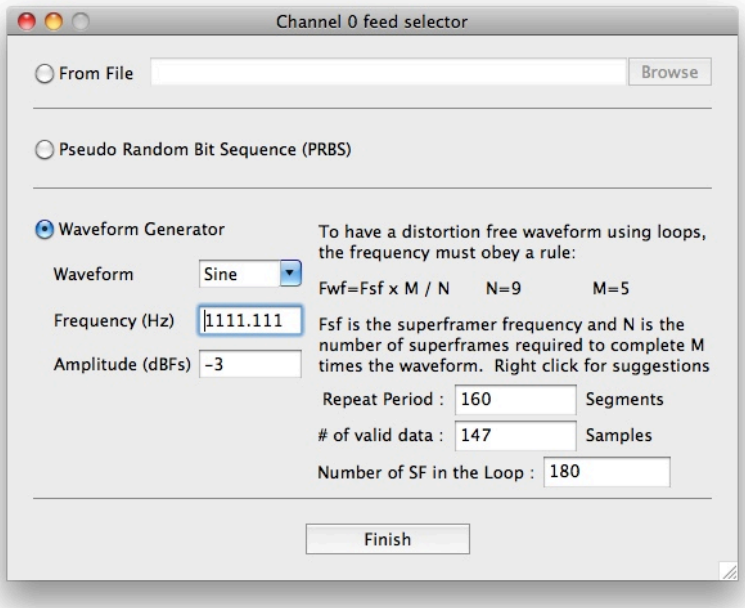
Even if the procedure to compute the number of superframes to put in a loop is relatively simple, ScriptBuilder is making it straightforward. The Source Selector window will indicate the minimum number of superframe to put in the loop, taking into account the protocol in use, the channel rate, the sampling rate and the tone frequency.

It will also indicate if the loop duration exceeds the maximum 1100 superframes allowed in a script.

With our example, the channel feed selector panel will give immediately the 20 superframe repeat period.



With another frequency like 1111.111111 Hz, the repeat is significantly longer : 180 superframes.



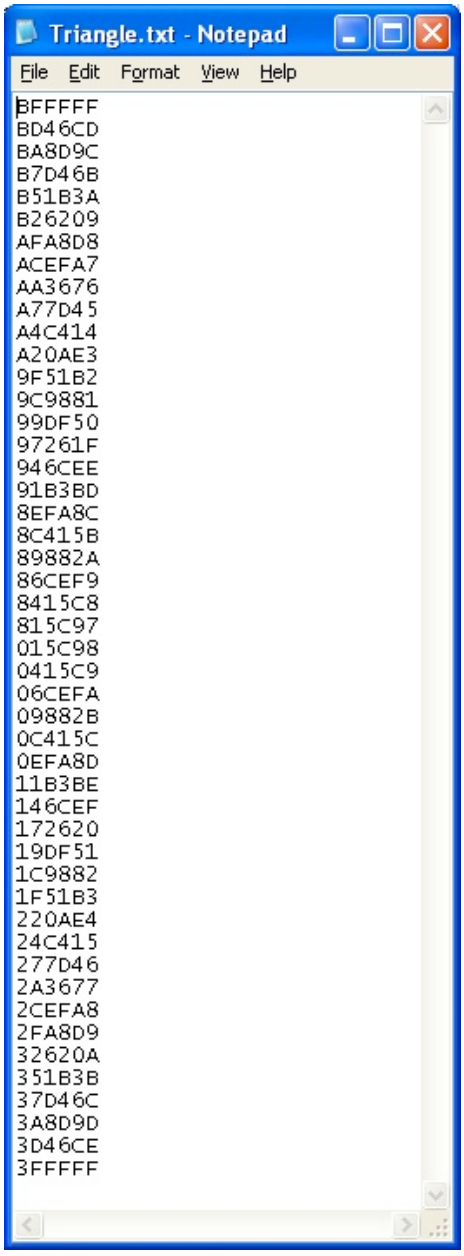
4. From file generation

Instead of using the Traffic Generator to build the sample values, we also have the possibility to use the sample values contained in a text file. The Traffic Generator will read the samples from the file as long as new superframes are generated. If the files contains a limited number of samples, the file content will be repeated over and over.

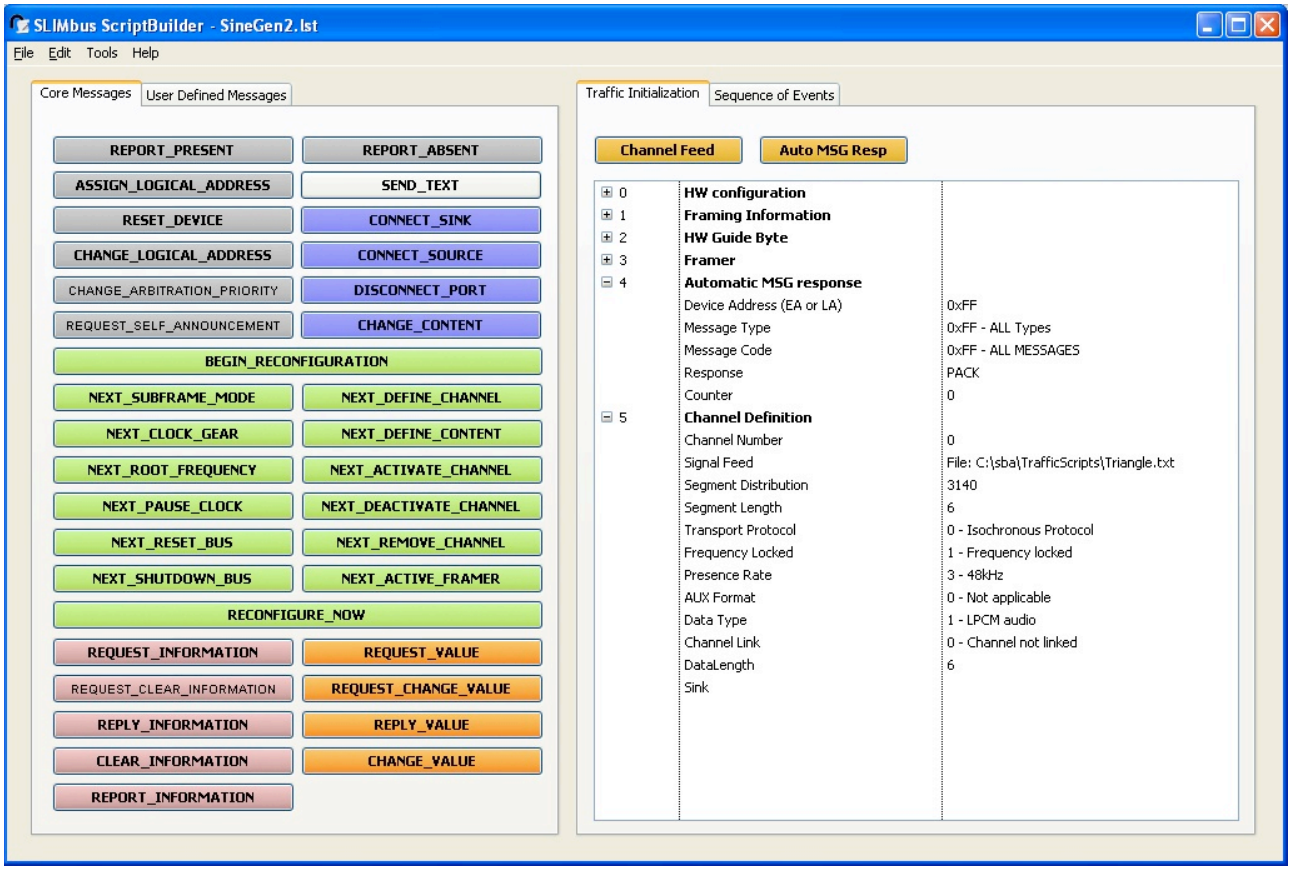
The same constraints as for the sine wave generation will apply to the number of samples read from the file. If we want a 1 ms period in a 48 Khz channel, we will need to provide 48 samples and so on.

The samples are provided in hexadecimal value with the offset sign&magnitude coding).

The following example shows the values for a sawtooth signal with -6 dBFS peak level.



We need change the channel feed in ScriptBuilder to get the file content into the data channel.



The “Signal Feed” parameter value can be changed manually by typing “File:” followed by the file path and file name or the Source Selector panel can be used to browse and select a file. When this script is played, the Protocol Analyzer will show the following plot.