

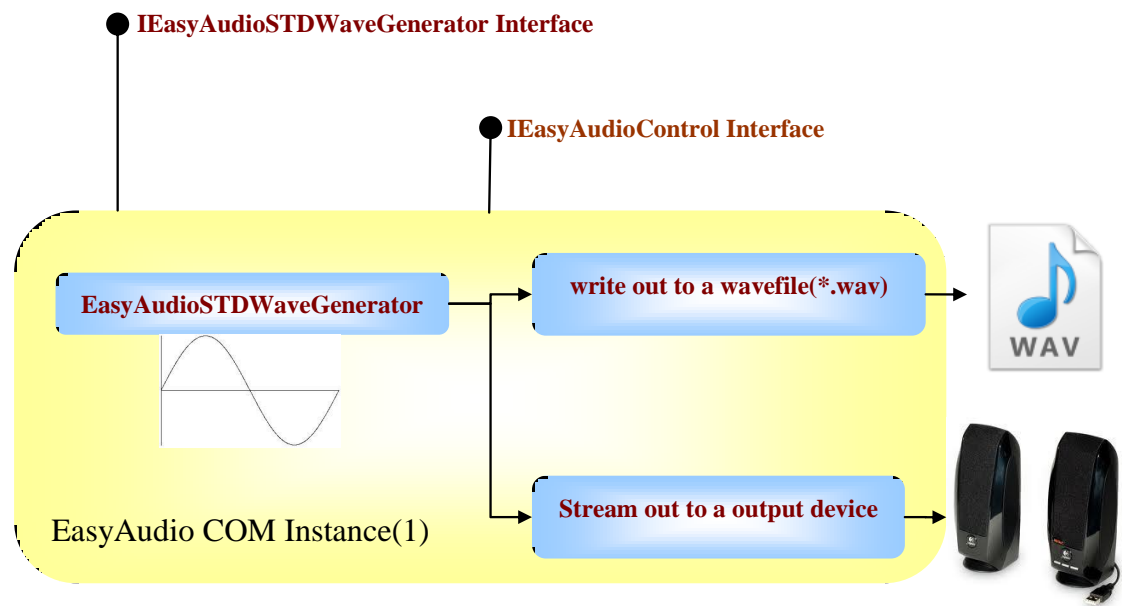
EasyAudio Programming Guide

Applied EasyAudio sine signal generator with source name

“EasyAudioSTDWaveGenerator” and IEasyAudioSTDWaveGenerator interface

Content

Content	1
Introduction.....	3
Purpose	3
EasyAudio Standard Waveform Generator Mechanism	3
EasyAudioSTDWaveGenerator’s Audio Formats	4
EasyAudioSTDWaveGenerator Requirement	6
Distribution.....	6
Sample Code.....	6
Project: EasyAudioATLConsoleApp.....	6
Sample Code Output Results	8
EasyAudioFormat(16000Hz-16Bits-1Ch)_Signal(Sine_300Hz)_Out(WF_SPK).wav	8
EasyAudioFormat(16000Hz-16Bits-1Ch)_Signal(Sine_800Hz)_Out(WF_SPK).wav	9
EasyAudioFormat(48000Hz-16Bits-1Ch)_Signal(Sine_4000Hz)_Out(WF_SPK).wav	10
EasyAudioFormat(44100Hz-16Bits-1Ch)_Signal(Sine_2000Hz)_Out(WF_SPK).wav	12
EasyAudioFormat(96000Hz-16Bits-1Ch)_Signal(Sine_5000Hz)_Out(WF_SPK).wav	13
EasyAudio for Speaker test	15
References.....	15
IEasyAudioSTDWaveGenerator Interface	15
IEasyAudioControl Interface	16
Function: AppSTDWaveGen2WF_SPK()	17
Other References	18
Acknowledgments	18



Juiwen Hsu@2013-0410

Introduction

Purpose

With Audio device test, we usually need different frequency of sine wave to be the standard signal to send into a device.

EasyAudio provide an interface **IEasyAudioSTDWaveGenerator** to generate standard signal into audio wave stream. It can output to a wave file and output to a device (speaker) at the same time.

EasyAudio Standard Waveform Generator Mechanism

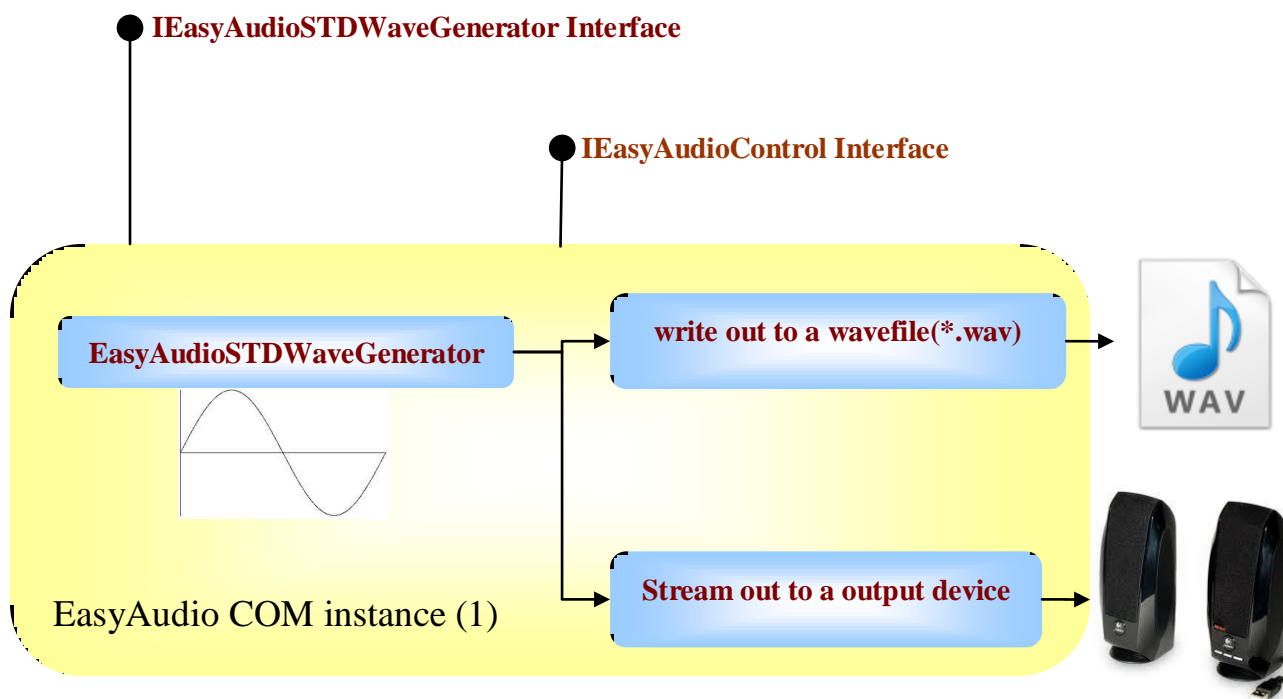
EasyAudioSTDWaveGenerator provide a simple way to generate standard signal (for example sine wave) with specify different properties for the signal wave.

The client can control the signal wave's properties via **IEasyAudioSTDWaveGenerator interface**.

With **IEasyAudioSTDWaveGenerator interface**, the client application can control below properties:

Signal type, SignalFrequency, SignalAmplitude and AmplitudeOffset.

More details about the **IEasyAudioSTDWaveGenerator interface**, please read the Reference section in the document.



EasyAudioSTDWaveGenerator's Audio Formats

In current version, **EasyAudioSTDWaveGenerator** supports audio formats as below:

Audio Input Device Format Count=18

- [0].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 16000 Hz, Bit Rate= 256000 bps,
- [1].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 24000 Hz, Bit Rate= 384000 bps,
- [2].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 32000 Hz, Bit Rate= 512000 bps,
- [3].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 48000 Hz, Bit Rate= 768000 bps,
- [4].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 96000 Hz, Bit Rate= 1536000 bps,
- [5].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 8000 Hz, Bit Rate= 128000 bps,
- [6].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 44100 Hz, Bit Rate= 705600 bps,
- [7].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 22050 Hz, Bit Rate= 352800 bps,
- [8].Wave Format=PCM, Channels= 1, Bits per Sample= 16 (bits), Sample Rate= 11025 Hz, Bit Rate= 176400 bps,
- [9].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 16000 Hz, Bit Rate= 512000 bps,
- [10].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 24000 Hz, Bit Rate= 768000 bps,
- [11].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 32000 Hz, Bit Rate= 1024000 bps,
- [12].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 48000 Hz, Bit Rate= 1536000 bps,
- [13].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 96000 Hz, Bit Rate= 3072000 bps,

- [14].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 8000 Hz, Bit Rate= 256000 bps,
 - [15].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 44100 Hz, Bit Rate= 1411200 bps,
 - [16].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 22050 Hz, Bit Rate= 705600 bps,
 - [17].Wave Format=PCM, Channels= 2, Bits per Sample= 16 (bits), Sample Rate= 11025 Hz, Bit Rate= 352800 bps,
-

You can specify a audio format via **IEasyAudioControl** Interface's `SetAudioSource()` method.

EasyAudioSTDWaveGenerator Requirement

EasyAudio provides [EasyAudioSTDWaveGenerator](#) in version 20.11.9.8 or latter.
EasyAudioSTDWaveGeneratorDRU.dll and EasyAudioDRU.dll must in the same folder.

Distribution

The client applications need to have below components in run-time.

EasyAudioSTDWaveGeneratorDRU.dll

EasyAudioDRU.dll

Sample Code

Project: EasyAudioATLConsoleApp

The sample code for EasyAudioSTDWaveGenerator is in the project: **EasyAudioATLConsoleApp**, you can find it in the EasyAudio release package.

The main example is inside CEasyAudioHelper class's member function:

```
HRESULT CEasyAudioHelper:: AppSTDWaveGen2WF_SPK (EasyAudio::IEasyAudioControl *pEACX)
```

In the console mode main() function, we it as

```
eat.AppSTDWaveGen2WF_SPK(eat.GetEasyAudioControl());//STDWaveGen in and output to speaker
```

Below is the configuration in the console's main() function to work with **AppSTDWaveGen2WF_SPK()**;

```
#ifdef DEMO_STD_WAVE_GENERATOR

    //AppSTDWaveGen2WF_SPK: Play the standard wave generator (EasyAudioSTDWaveGenerator) to wavefile and speaker at the same time.

    //AID audio waveform format =====

    eat.m_nChannels=1; // EasyAudio standard wave generator support 1 channel.
    eat.m_wBitsPerSample=16;
    eat.m_nSamplesPerSec=16000;
    eat.m_AIDVolume=1.0;
```

```

//AOD properties =====
eat.m_AODVolume=0;// 0 means full value

//WF Out =====

eat.m_DelaySamplingCount=000;/* avoid samples after play */
eat.m_SamplingCount=24000;/* samples to get */

//

eat.SetWaveFileSource(_T(""));

//=====

eat.SetMicDeviceName(_T("EasyAudioSTDWaveGenerator"));// put EasyAudioSTDWaveGenerator name here.
eat.SetAudioOutDeviceName(_T("Default DirectSound Device"));//the output device name
eat.SetWaveOutFileName(_T("EasyAudioFormat(16000Hz-16Bits-1Ch)_Signal(Sine_800Hz)_Out(WF_SPK).wav")); //det default
wave ouput filename

eat.pSWG->SetSTDWaveFormat(0, 800, 0.5, 0.0);// signal type, SignalFrequency, SignalAmplitude, AmplitudeOffset
eat.AppSTDWaveGen2WF_SPK(eat.GetEasyAudioControl());//STDWaveGen in and output to speaker

//=====

eat.SetMicDeviceName(_T("EasyAudioSTDWaveGenerator"));// put EasyAudioSTDWaveGenerator name here.
eat.SetAudioOutDeviceName(_T("Default DirectSound Device"));//the output device name
eat.SetWaveOutFileName(_T("EasyAudioFormat(16000Hz-16Bits-1Ch)_Signal(Sine_300Hz)_Out(WF_SPK).wav")); //det default
wave ouput filename

eat.pSWG->SetSTDWaveFormat(0, 300, 0.5, 0.25);
eat.AppSTDWaveGen2WF_SPK(eat.GetEasyAudioControl());//STDWaveGen in and output to speaker

//=====

eat.m_nSamplesPerSec=48000;
eat.SetMicDeviceName(_T("EasyAudioSTDWaveGenerator"));// put EasyAudioSTDWaveGenerator name here.
eat.SetAudioOutDeviceName(_T("Default DirectSound Device"));//the output device name
eat.SetWaveOutFileName(_T("EasyAudioFormat(48000Hz-16Bits-1Ch)_Signal(Sine_4000Hz)_Out(WF_SPK).wav")); //det
default wave ouput filename

eat.pSWG->SetSTDWaveFormat(0, 4000, 0.5, 0.0);
eat.AppSTDWaveGen2WF_SPK(eat.GetEasyAudioControl());//STDWaveGen in and output to speaker

//=====

eat.m_nSamplesPerSec=44100;
eat.SetMicDeviceName(_T("EasyAudioSTDWaveGenerator"));// put EasyAudioSTDWaveGenerator name here.
eat.SetAudioOutDeviceName(_T("Default DirectSound Device"));//the output device name
eat.SetWaveOutFileName(_T("EasyAudioFormat(44100Hz-16Bits-1Ch)_Signal(Sine_2000Hz)_Out(WF_SPK).wav")); //det
default wave ouput filename

```

```
eat.pSWG->SetSTDWaveFormat(0, 2000, 1.0, 0.0);

eat.AppSTDWaveGen2WF_SPK(eat.GetEasyAudioControl()); //STDWaveGen in and output to speaker

#endif
```

Sample Code Output Results

EasyAudioFormat(16000Hz-16Bits-1Ch)_Signal(Sine_300Hz)_Out(WF_SPK).wav

Audio Format:

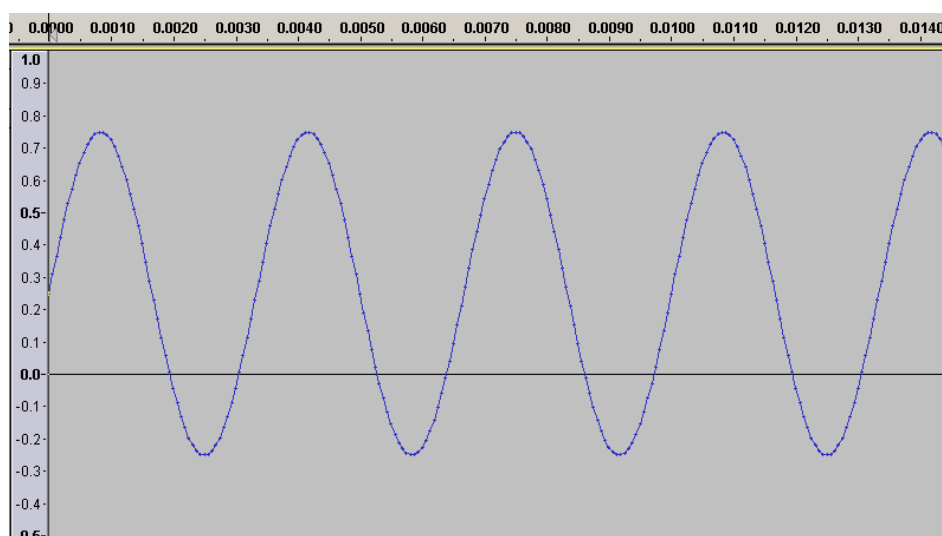
Channels	Bits per Sample(bits)	Sample Rate (Hz)	Bit Rate (bps)
1	16	16000	256000

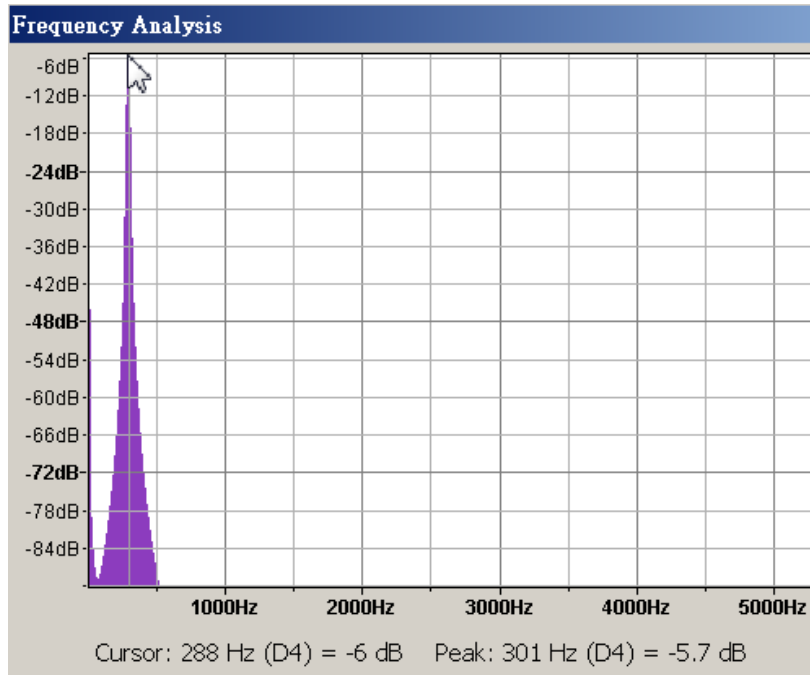
Signal Format:

Signal Type= Sine Wave, SignalFrequency= 300 Hz, SignalAmplitude= 0.5, AmplitudeOffset= 0.25

EasyAudio Example code:

```
eat.pSWG->SetSTDWaveFormat(0, 300, 0.5, 0.25); // signal type, SignalFrequency, SignalAmplitude, AmplitudeOffset
```





EasyAudioFormat(16000Hz-16Bits-1Ch)_Signal(Sine_800Hz)_Out(WF_SPK).wav

Audio Format:

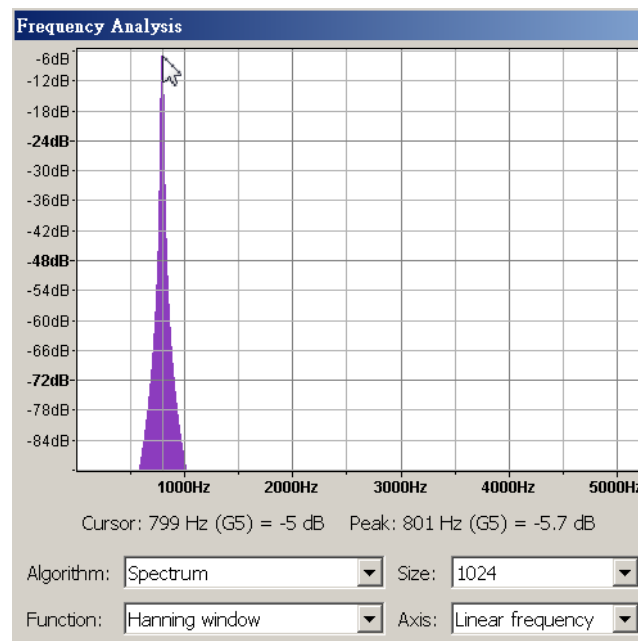
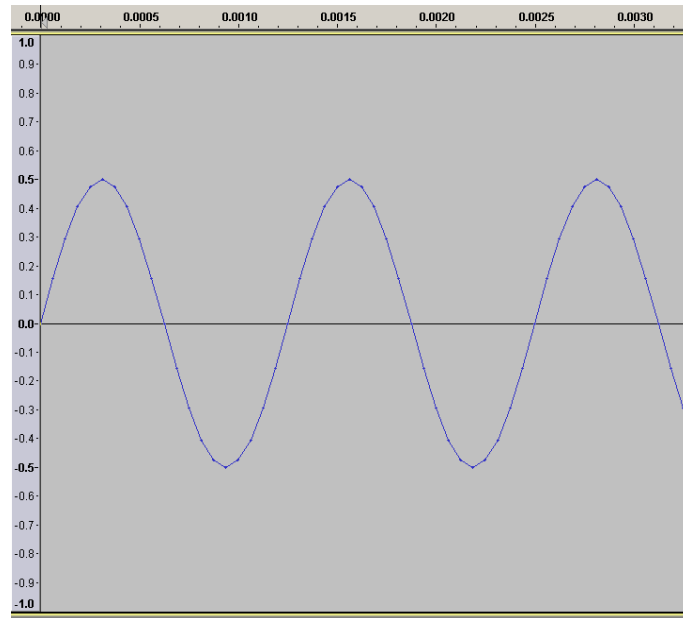
Channels	Bits per Sample(bits)	Sample Rate (Hz)	Bit Rate (bps)
1	16	16000	256000

Signal Format:

Signal Type= Sine Wave, SignalFrequency= 800 Hz, SignalAmplitude= 0.5, AmplitudeOffset= 0.0,

EasyAudio Example code:

```
eat.pSWG->SetSTDWaveFormat(0, 800, 0.5, 0.0);// signal type, SignalFrequency, SignalAmplitude, AmplitudeOffset
```



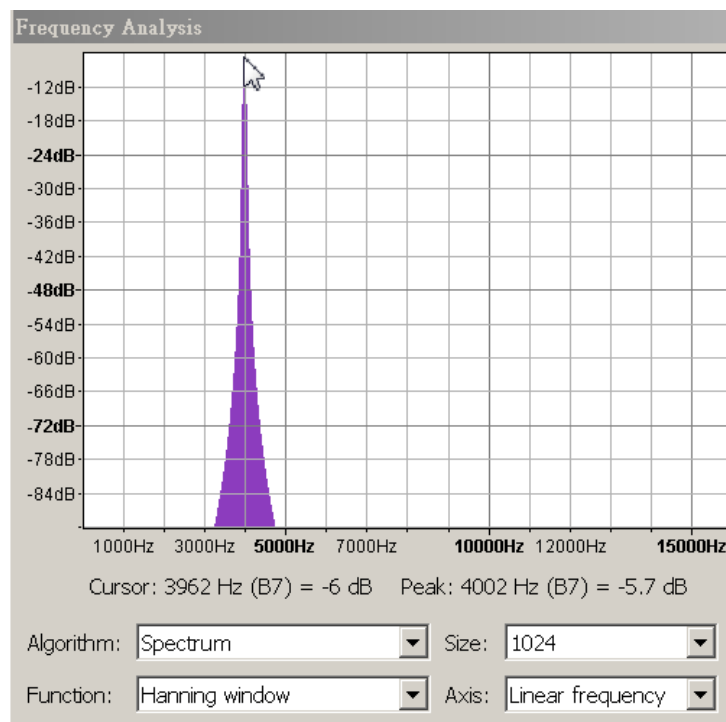
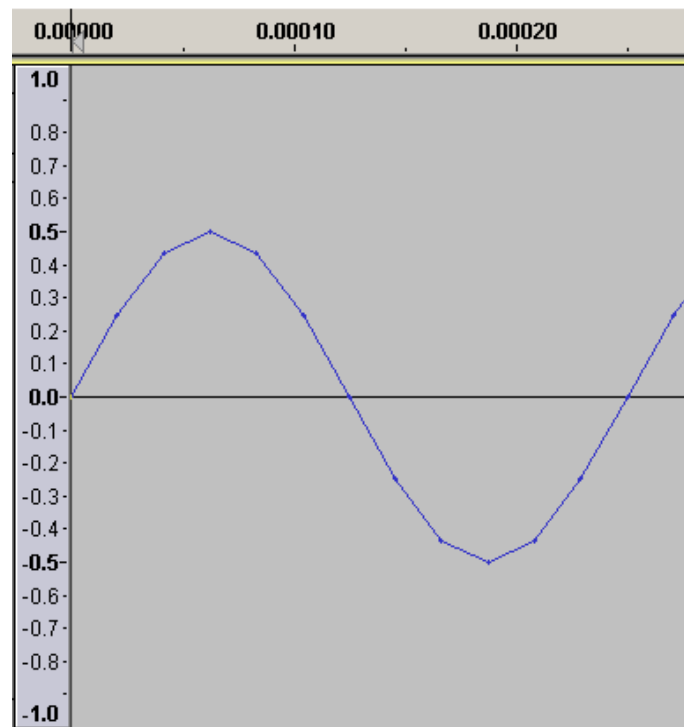
EasyAudioFormat(48000Hz-16Bits-1Ch)_Signal(Sine_4000Hz)_Out(WF_SPK).wav

Audio Format:

Channels	Bits per Sample(bits)	Sample Rate (Hz)	Bit Rate (bps)
1	16	16000	768000

Signal Format:

Signal Type= Sine Wave, SignalFrequency=4000 Hz, SignalAmplitude= 0.5, AmplitudeOffset= 0.0



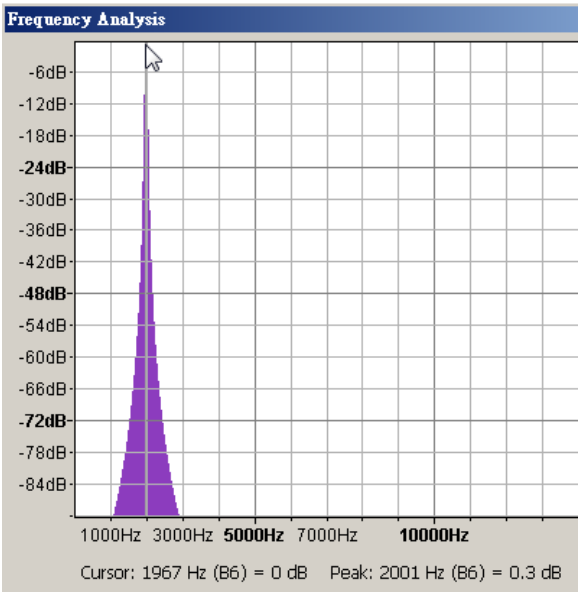
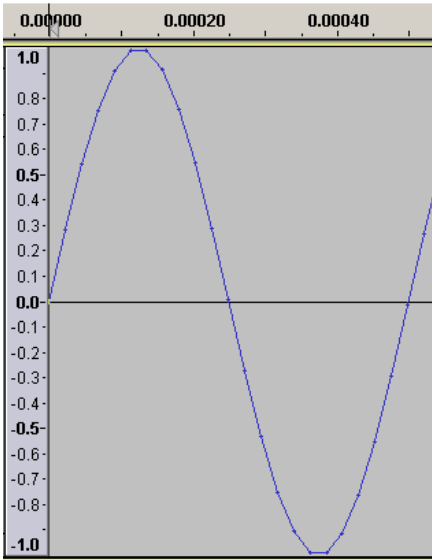
EasyAudioFormat(44100Hz-16Bits-1Ch)_Signal(Sine_2000Hz)_Out(WF_SPK).wav

Audio Format:

Channels	Bits per Sample(bits)	Sample Rate (Hz)	Bit Rate (bps)
1	16	44100	705600

Signal Format:

Signal Type= Sine Wave, SignalFrequency=2000 Hz, SignalAmplitude= 1.0, AmplitudeOffset= 0.0



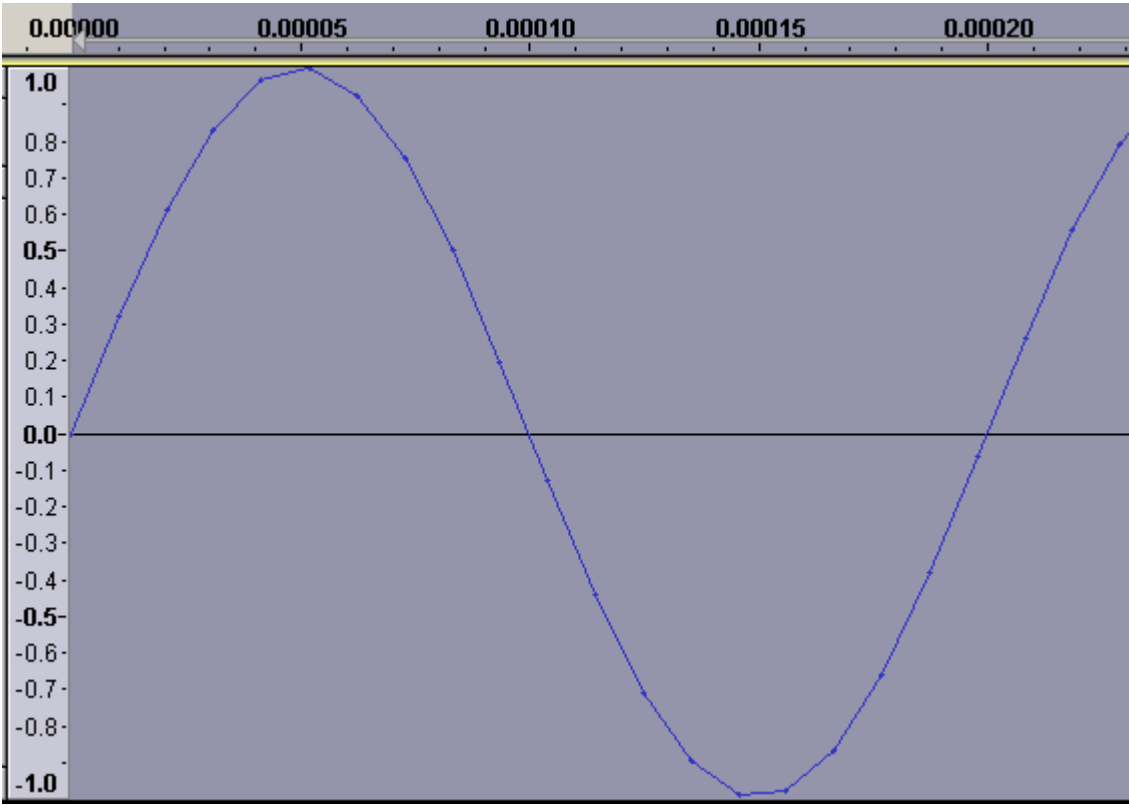
EasyAudioFormat(96000Hz-16Bits-1Ch)_Signal(Sine_5000Hz)_Out(WF_SPK).wav

Audio Format:

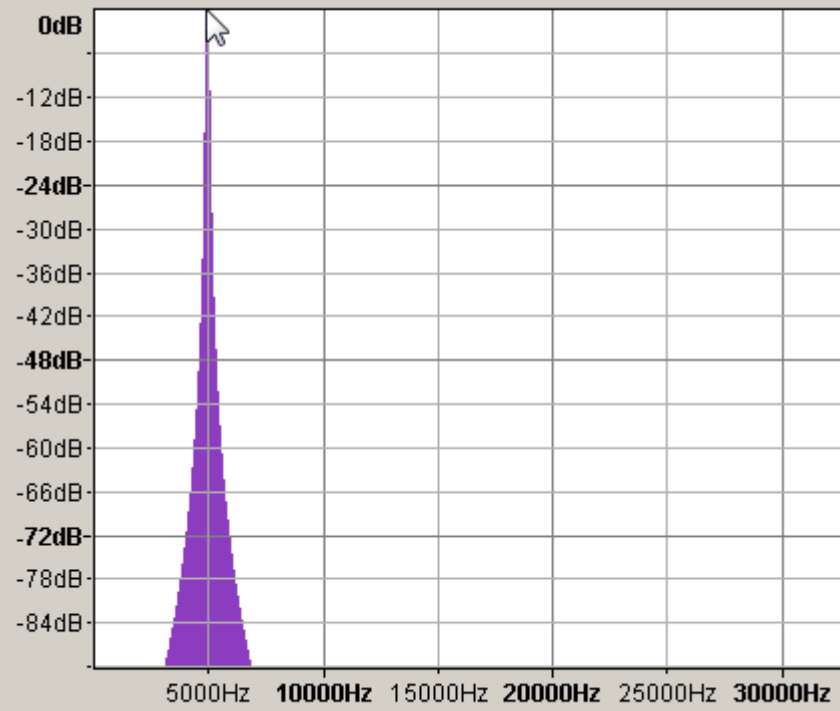
Channels	Bits per Sample(bits)	Sample Rate (Hz)	Bit Rate (bps)
1	16	96000	1536000

Signal Format:

Signal Type= Sine Wave, SignalFrequency=5000 Hz, SignalAmplitude= 1.0, AmplitudeOffset= 0.0

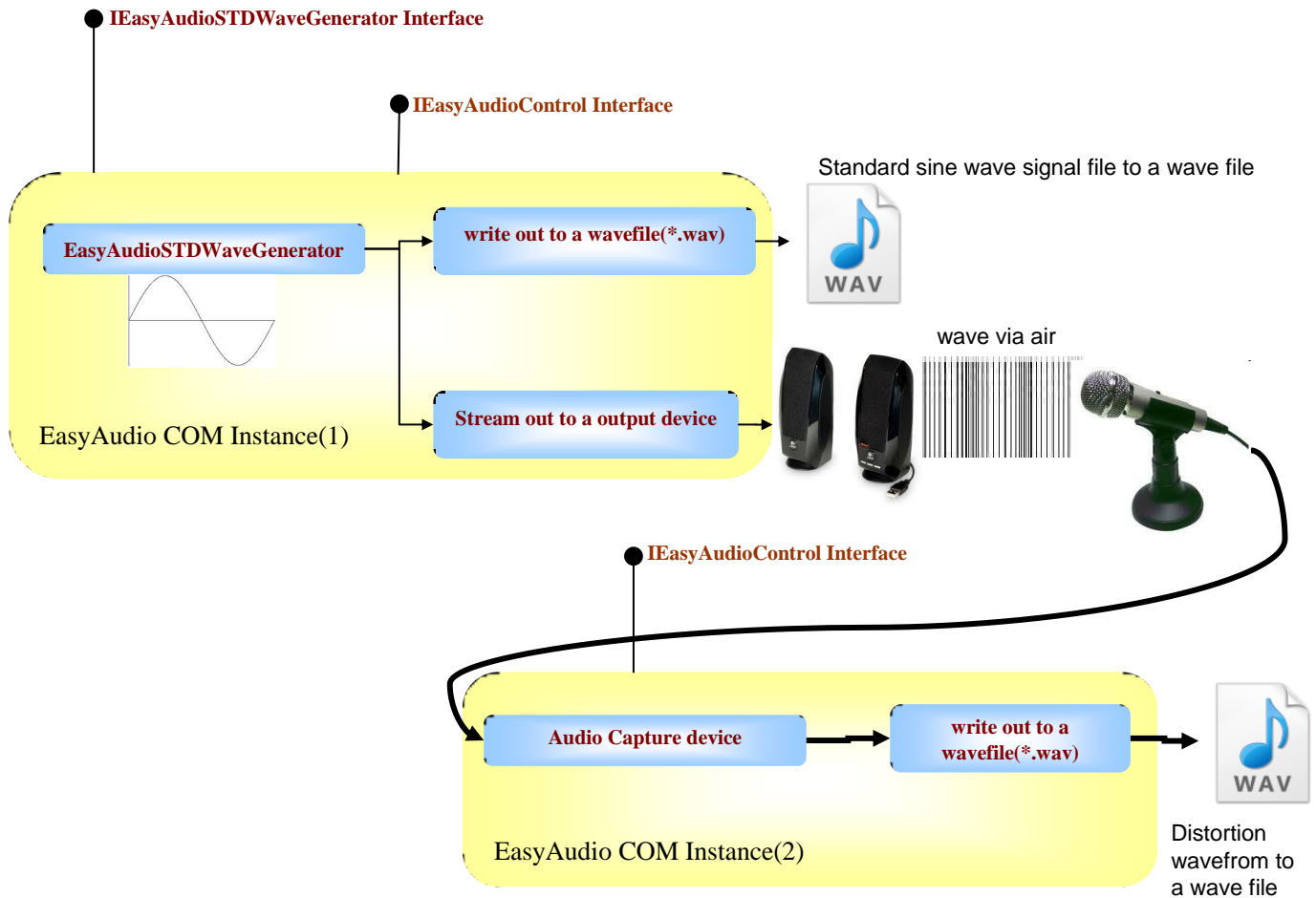


Frequency Analysis



Cursor: 4893 Hz (D#8) = -2 dB Peak: 5004 Hz (D#8) = 0.3 dB

EasyAudio for Speaker test



References

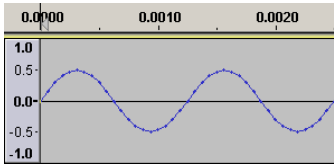
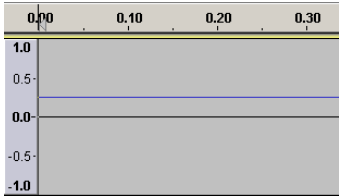
IEasyAudioSTDWaveGenerator Interface

With IEasyAudioSTDWaveGenerator Interface you can specify the waveform format for the standard waveform, for example sine wave.

IEasyAudioSTDWaveGenerator exports below methods for clinet application:

```
HRESULT SetSTDWaveFormat(UINT SignalType, double SignalFrequency, double SignalAmplitude, double AmplitudeOffset);
```

SignalType: Signal type can be one of below.

SignalType	Description	Example:
0	Sine Wave	Sine Wave with SignalAmplitude=0.5, SignalFrequency=800, AmplitudeOffset=0.0 
1	Just a DC	Example: DC with AmplitudeOffset=0.25. (25%) 

SignalFrequency: Signal frequency in Hz. For example **SignalFrequency**= 1000 means 1 KHz.

SignalAmplitude: the value is a ratio of the signal’s amplitude, the range of the value is from 0~1.0.

AmplitudeOffset: the ratio of the Amplitude offset, the range of the value is from 0~1.0.

IEasyAudioControl Interface

IEasyAudioControl Interface exports below methods for clinet application:

```
HRESULT SetAudioSource(LPTSTR AIS, WORD nChannels,WORD wBitsPerSample, DWORD nSamplesPerSec, double AIDVolume);
HRESULT SetAudioOutDevice(LPTSTR AOD, long OutVol);
HRESULT SetAudioOutWaveFile(LPTSTR AOWF, ULONG DelaySamplingCount, ULONG SamplingCount);
HRESULT Play(BOOL bWaitCaptureTime);
HRESULT Stop(void);
BOOL IsPlaying(void);
int AIDGetCount(void);
int AODGetCount(void);
```

The sample code please refer function AppSTDWaveGen2WF_SPK() in EasyAudioATLConsoleApp project.

Function: AppSTDWaveGen2WF_SPK()

```
//AppSTDWaveGen2WF_SPK: Play the standard wave generator (EasyAudioSTDWaveGenerator) to wavefile and speaker at the
same time.
HRESULT CEasyAudioHelper::AppSTDWaveGen2WF_SPK(EasyAudio::IEasyAudioControl *pEACX)
{
m_pLogMain->Line(80);
m_pLogMain->Out(_T("[Test Case]: Play the standard wave generator (EasyAudioSTDWaveGenerator) to wavefile and speaker
at the same time. "));
m_pLogMain->Out(_T(" Example Function= CEasyAudioHelper::AppSTDWaveGen2WF_SPK()"));
m_pLogMain->Line(80);
m_pLogMain->Out(DumpConfiguration());
// =====
// Audio Source: USB audio Device
hr= pEACX->SetAudioSource(m_pMicDeviceName,m_nChannels,m_wBitsPerSample,m_nSamplesPerSec,m_AIDVolume);
if(FAILED(hr))
{
    m_pLogError->Out(_T("error: Audio Input Device not found in the system: %s"),m_pMicDeviceName);
    return hr;
}
// Audio Out: Speaker device: No specified
hr= pEACX->SetAudioOutDevice(m_pAODName,m_AODVolume); // Full volume is 0, and -,000 is silence.
if(FAILED(hr))
{
    m_pLogError->Out(_T("error: SetAudioOutDevice(): %s"));
    return hr;
}
hr= pEACX->SetAudioOutWaveFile(m_pWaveOutFileName,m_DelaySamplingCount,m_SamplingCount); //UAC: USB audio device
WF: wavefile,
// =====
//          (DelaySamplingCount)          (SamplingCount)
//[Play Audio]<----->[Auto Start Capture]<----->[Auto Stop Capture]<---->[Stop play]
if(FAILED(hr))
{
    m_pLogError->Out(_T("error: SetAudioOutWaveFile(%s)"),m_pWaveOutFileName);
}
//=====
hr=RunWith(pEACX,TRUE);
if(FAILED(hr))
```

```
{  
    m_pLogError->Out(_T("error: AppACS2WF("));  
}  
return hr;  
}
```

Other References

[1]. EasyAudio COM Library:

<https://docs.google.com/a/logitech.com/?tab=mo#folders/0B60SgTBmsbI8NzU5MDNjODQtMmJmNC00ZTVjLWI5NDgtZmRmOTBkMmNjZWZh>

[2]. EasyAudio Project Site: <https://sites.google.com/a/logitech.com/easyaudio/>

Acknowledgments

The following people took part in the development of EasyAudio and/or contributed to the creation of this document:

Yogi Lin