EasyAudio Programming Guide

Applied EasyAudio sine signal generator with source name “EasyAudioSTDWaveGenerator” and IEasyAudioSTDWaveGenerator interface

# Content

[Content 1](#_Toc353394587)

[Introduction 3](#_Toc353394588)

[Purpose 3](#_Toc353394589)

[EasyAudio Standard Waveform Generator Mechanism 3](#_Toc353394590)

[EasyAudioSTDWaveGenerator’s Audio Formats 4](#_Toc353394591)

[EasyAudioSTDWaveGenerator Requirement 6](#_Toc353394592)

[Distribution 6](#_Toc353394593)

[Sample Code 6](#_Toc353394594)

[Project: EasyAudioATLConsoleApp 6](#_Toc353394595)

[Sample Code Output Results 8](#_Toc353394596)

[EasyAudioFormat(16000Hz-16Bits-1Ch)\_Signal(Sine\_300Hz)\_Out(WF\_SPK).wav 8](#_Toc353394597)

[EasyAudioFormat(16000Hz-16Bits-1Ch)\_Signal(Sine\_800Hz)\_Out(WF\_SPK).wav 9](#_Toc353394598)

[EasyAudioFormat(48000Hz-16Bits-1Ch)\_Signal(Sine\_4000Hz)\_Out(WF\_SPK).wav 10](#_Toc353394599)

[EasyAudioFormat(44100Hz-16Bits-1Ch)\_Signal(Sine\_2000Hz)\_Out(WF\_SPK).wav 12](#_Toc353394600)

[EasyAudioFormat(96000Hz-16Bits-1Ch)\_Signal(Sine\_5000Hz)\_Out(WF\_SPK).wav 13](#_Toc353394601)

[EasyAudio for Speaker test 15](#_Toc353394602)

[References 15](#_Toc353394603)

[IEasyAudioSTDWaveGenerator Interface 15](#_Toc353394604)

[IEasyAudioControl Interface 16](#_Toc353394605)

[Function: AppSTDWaveGen2WF\_SPK() 17](#_Toc353394606)

[Other References 18](#_Toc353394607)

[Acknowledgments 18](#_Toc353394608)



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**

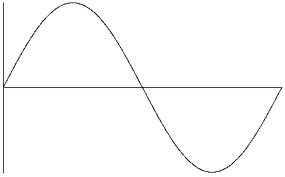
**write out to a wavefile(\*.wav)**

**Stream out to a output device**

**IEasyAudioSTDWaveGenerator Interface**

**IEasyAudioControl Interface**

EasyAudio COM instance (1)



.

# 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:

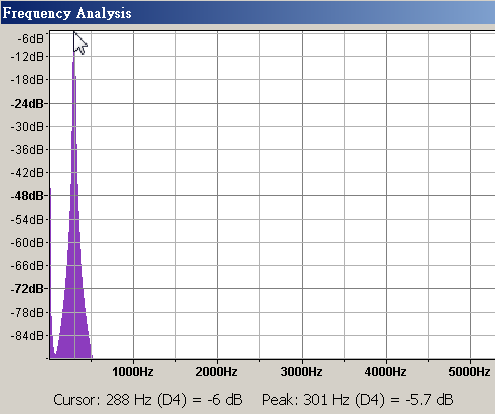
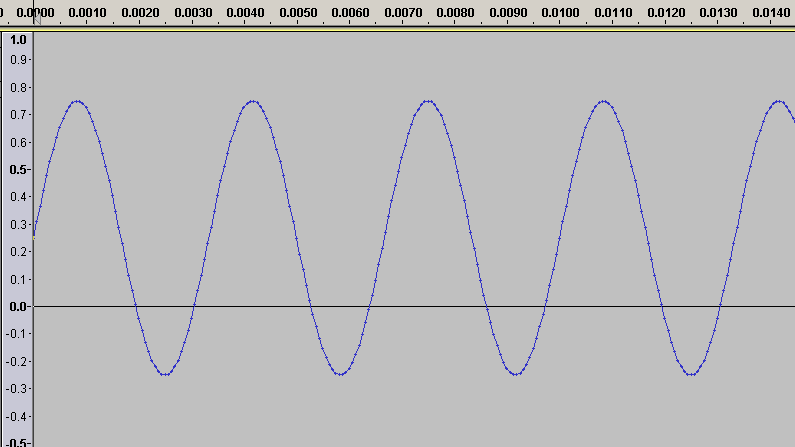
|  |  |  |  |
| --- | --- | --- | --- |
| Channels | Bits per Sample(bits) | Sample Rate (Hz) | Bit Rate (bps) |
| 1 | 16 | 16000 | 256000 |

#### 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:

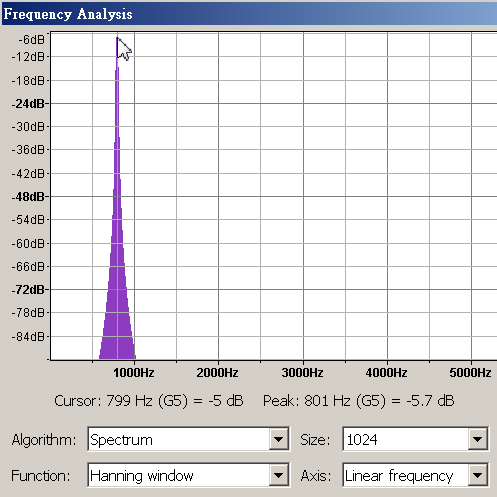
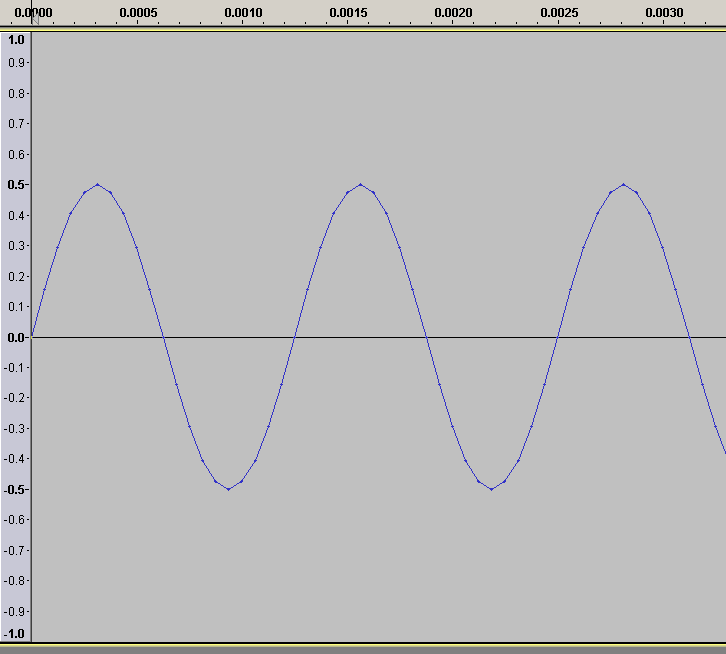
|  |  |  |  |
| --- | --- | --- | --- |
| Channels | Bits per Sample(bits) | Sample Rate (Hz) | Bit Rate (bps) |
| 1 | 16 | 16000 | 256000 |

#### 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 |



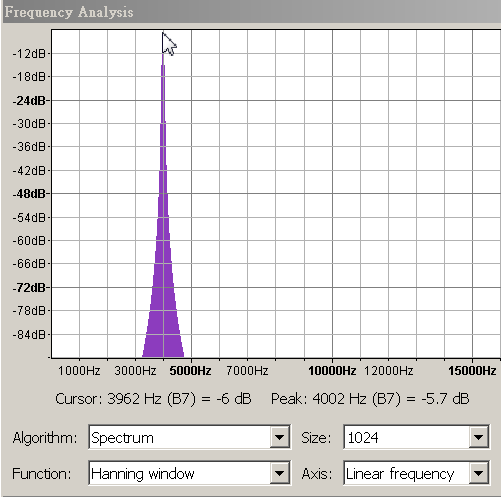
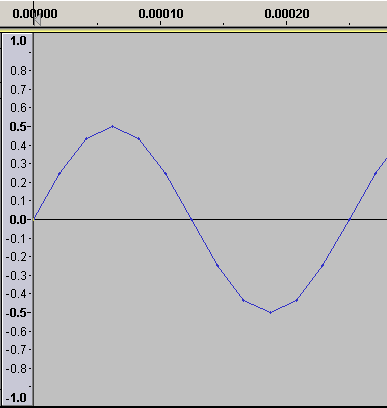
### EasyAudioFormat(48000Hz-16Bits-1Ch)\_Signal(Sine\_4000Hz)\_Out(WF\_SPK).wav

#### 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



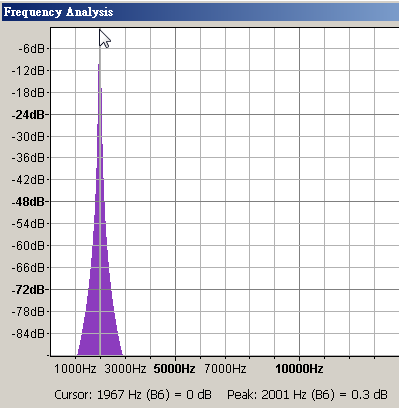
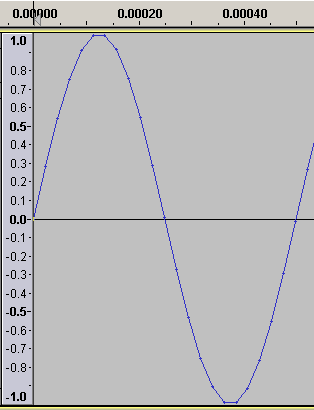
### EasyAudioFormat(44100Hz-16Bits-1Ch)\_Signal(Sine\_2000Hz)\_Out(WF\_SPK).wav

#### 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



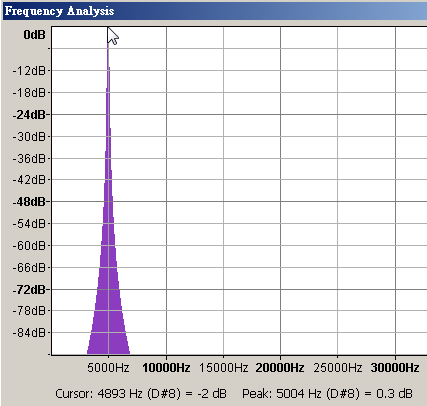
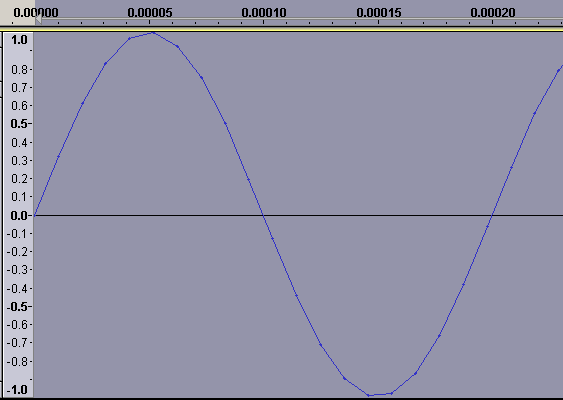
### EasyAudioFormat(96000Hz-16Bits-1Ch)\_Signal(Sine\_5000Hz)\_Out(WF\_SPK).wav

#### 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



# 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/0B60SgTBmsbI8NzU5MDNjODQtMmJmNC00ZTVjLWI5NDgtZmRmOTBkMmNjZWRh](https://docs.google.com/a/logitech.com/?tab=mo%23folders/0B60SgTBmsbI8NzU5MDNjODQtMmJmNC00ZTVjLWI5NDgtZmRmOTBkMmNjZWRh)

[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