AWE Library

Generated by Doxygen 1.8.6

Fri Dec 13 2019 06:53:24

Contents

1 Main Page

Description

This document describes the API for accessing the Audio Weaver (AWE) Run-time core. The core executes a signal processing algorithm (or layout) defined by an Audio Weaver Binary (AWB) file. To do this, the core must be initialized, an AWB must be loaded, audio data must be passed to the core, and the layout must be executed or "pumped". Upon completion of the pump, any output data will be available. An example application using this library is included. Also included is code that relates this API to the basic DSP Concepts API.

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 Module Documentation

4.1 AWELibrary

Classes

- union _Sample

  Union of all possible sample types.
- class CAWELib

  Interface for the AudioWeaver library.

Macros

- #define MAKE_ADDRESS(id, idx) (((id) << 12) | (idx))

  Create a virtual address.
- #define **MAKE_WIRE_ADDRESS**(id) MAKE_ADDRESS(id, 3)
- #define **MAKE_LAYOUT_ADDRESS**(id) MAKE_ADDRESS(id, 0)
- #define **UINT32** unsigned int
- #define **INT32** int
- #define **UINT64** unsigned long long
- #define **MAX_COMMAND_BUFFER_LEN** (4096 + 9)

Typedefs

- typedef union _Sample Sample

  Union of all possible sample types.

Functions

- CAWELib * AWELibraryFactory ()

  Create an instance of the library object.

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 #define MAKE_ADDRESS(*id, idx*) (((id) << 12) | (idx))

Create a virtual address.

4.1.3 Typedef Documentation

4.1.3.1 typedef union _Sample Sample

Union of all possible sample types.

4.1.4 Function Documentation

4.1.4.1 CAWELib* AWELibraryFactory ( )

Create an instance of the library object.

Returns

the instance

4.2 Errors

Macros

- #define E_SUCCESS (0)

  OK result.
- #define E_HEAP_INDEX_RANGE (-1)

  Attempt to reference a non-existent heap.
- #define E_MALLOC_SIZE_TOO_BIG (-2)

  Attempt to allocate more storage with awe_fwMalloc() than exists.
- #define E_SCRATCH_ALLOC_SIZE_TOO_BIG (-3)

  Attempt to allocate more storage with awe_fwMallocScratch() than exists.
- #define E_CONSTRUCTOR_ARGUMENT_COUNT (-4)

  A constructor was called with the wrong number of arguments.
- #define E_CLASS_INDEX_RANGE (-5)

  Attempt to reference a non-existent class.
- #define E_CLASS_NOT_FOUND (-6)

  Can't find the specified class.
- #define E_MODULE_ALREADY_OWNED (-7)

  Attempt to bind a module instance to a layout that is already owned by a layout.
- #define E_ASSIGN_ADDRESS_OUT_OF_RANGE (-8)

  Attempt to assign outside the bounds of the heap.
- #define E_MODULE_NOT_WIRE (-9)

  A wire input given to a module constructor is not a wire.
- #define E_INPUTS_MUST_MATCH_OUTPUTS (-10)

  Many modules require the number of inputs and outputs to be the same.
- #define E_INPUTS_MUST_BE_SAME_PINTYPE (-11)

  Many modules require that all inputs be of the same pintype.
- #define E_MUST_HAVE_INPUTS (-12)

  Many modules require at leat one input.
- #define E_MUST_HAVE_OUTPUTS (-13)

  Many modules require at leat one output.
- #define E_INPUTS_MUST_MATCH_CORRESPONDING_OUTPUTS (-14)

  Many modules require the input and output types to match in pairs.
- #define E_NOT_MODULE (-15)

  The module given is not a module.
- #define E_INPUT_OUTPUT_COUNT (-16)

  Module with fixed inputs/outputs has wrong count.
- #define E_PARAMETER_ERROR (-17)

  Module arguments have an error.
- #define E_NO_MORE_OBJECTS (-18)

  awe_fwGetFirstObject() or awe_fwGetNextObject() reached end of object chain.
- #define E_NOT_OBJECT_POINTER (-19)

  Object pointer given to awe_fwGetNextObject() does not point to a module instance.
- #define E_NOT_INPUT_PIN (-20)

  Object pointer given to ClassInputWire_Constructor() is not an input pin.
- #define E_IOPIN_IN_USE (-21)

  The I/O pin given to ClassXXWire_Constructor() is already in use.
- #define E_PIN_TYPES_NOT_COMPATIBLE (-22)

  The wire and I/O pin type are not compatible.
- #define E_PIN_SIZES_NOT_COMPATIBLE (-23)

The wire and I/O pin sizes are not compatible.

- #define E_NOT_OUTPUT_PIN (-24)

  Object pointer given to ClassOutpuWire_Constructor() is not an output pin.
- #define E_NO_MORE_IOPINS (-25)

  awe_fwGetFirstIO() or awe_fwGetNextIO() reached end of object chain.
- #define E_NO_LAYOUTS (-26)

  Master pump function found no layouts to pump.
- #define E_MUST_HAVE_ONE_OUTPUT (-27)

  For modules with a hardwired single output (like RMS).
- #define E_OUTPUT_MUST_BE_SINGLE_VALUE (-28)

  For modules that need an output with 1 single output value (like RMS).
- #define E_INCOMPATIBLE_BLOCK_SIZES (-29)

  For containers, the contained module required sizes don't match.
- #define E_WIRE_INDEX_RANGE (-30)

  A write index in a vector is out of range.
- #define E_NO_EVENT_HANDLER (-31)

  There is no event handler for this module.
- #define E_AUDIO_ALREADY_STARTED (-32)

  Audio_Start() called with the audio already running.
- #define E_AUDIO_ALREADY_STOPPED (-33)

  Audio_Stop() called with the audio already stopped.
- #define E_COMMUNICATIONS_ERROR (-34)

  Framework communications failure.
- #define E_SALT_OBJECT_NOTFOUND (-35)

  Standalone tuning definitions.
- #define E_SALT_FIELD_RANGE (-36)

  Salt field range error.
- #define E_SALT_STATE_RANGE (-37)

  Salt state range error.
- #define E_NOT_IMPLEMENTED_IN_RS232 (-38)

  Attempt to pump for RS232.
- #define E_BADPACKET (-39)

  Bad packet - invalid argument.
- #define E_BADFILE (-40)

  Filename NULL.
- #define E_FILENAMELENGTH (-41)

  Filename length > 23.
- #define E_CANTCREATE (-42)

  Can't create file.
- #define E_CANTOPEN (-43)

  Can't open the specified file.
- #define E_NOSUCHFILE (-44)

  The specified file does not exist.
- #define E_IOERROR (-45)

  An error occurred accessing resource.
- #define E_FIND_FIRST_FILE_NOT_CALLED (-46)

  Find First File must be called First.
- #define E_NO_MORE_FILES (-47)

  No more files found.
- #define E_BAD_FILENAME (-48)

  The filename is not valid.

CFramework exception (null pointer?).

- #define E_MESSAGE_LENGTH_TOO_LONG (-75)

  Bad packet - message length is too long.
- #define E_CRC_ERROR (-76)

  Bad packet - CRC error.
- #define E_UNKNOWN_MESSAGE (-77)

  Bad packet - invalid command ID.
- #define E_MSG_TIMEOUT (-78)

  Message timed out (C6 only).
- #define E_OBJECT_ID_NOT_FOUND (-79)

  Object ID not found.
- #define E_PIN_ID_NOT_FOUND (-80)

  I/O pin not found.
- #define E_NOT_OBJECT (-81)

  Not a valid object.
- #define E_BAD_MEMBER_INDEX (-82)

  Member index is out of object bounds.
- #define E_CLASS_NOT_SUPPORTED (-83)

  Attempt to access member of wrong class type.
- #define E_PUMP_OVERRUN (-84)

  Attempt to pump when awe_fwTick() was not called.
- #define E_NOT_V5 (-85)

  Target is not a V5 target - legacy.
- #define E_NO_FRAMEWORK (-86)

  The framework does not exist.
- #define E_NO_CORE (-87)

  The specified core does not exist.
- #define E_IOPIN_TOO_MANY (-88)

  Too many wires bound to pin.
- #define E_WIRE_ALREADY_BOUND (-89)

  Attempt to bind a wire already bound.
- #define E_WIRES_NOT_SPECIFIED (-90)

  Required wires not specified.
- #define E_NOT_CREATED (-91)

  AWE instance not created.
- #define E_ALREADY_CREATED (-92)

  AWE instance already created.
- #define E_AUDIO_NOT_STARTED (-93)

  Can't pump, audio not started.
- #define E_LINKEDLIST_CORRUPT (-94)

  Module instance linked list is corrupted.
- #define E_INVALID_FILE (-95)

  The file content is invalid.
- #define E_MODULE_NOT_INITIALIZED (-96)

  The module was not initialized.

4.2.1 Detailed Description

4.2.2 Macro Definition Documentation

4.2.2.1 #define E_ALREADY_CREATED (-92)

AWE instance already created.

4.2.2.2 #define E_ARGUMENT_ERROR (-67)

Argument value invalid.

4.2.2.3 #define E_ASSIGN_ADDRESS_OUT_OF_RANGE (-8)

Attempt to assign outside the bounds of the heap.

4.2.2.4 #define E_AUDIO_ALREADY_STARTED (-32)

Audio_Start() called with the audio already running.

4.2.2.5 #define E_AUDIO_ALREADY_STOPPED (-33)

Audio_Stop() called with the audio already stopped.

4.2.2.6 #define E_AUDIO_NOT_STARTED (-93)

Can't pump, audio not started.

4.2.2.7 #define E_BAD_MEMBER_INDEX (-82)

Member index is out of object bounds.

4.2.2.8 #define E_CANTCREATE (-42)

Can't create file.

4.2.2.9 #define E_CANTOPEN (-43)

Can't open the specified file.

4.2.2.10 #define E_CLASS_INDEX_RANGE (-5)

Attempt to reference a non-existent class.

4.2.2.11 #define E_CLASS_NOT_FOUND (-6)

Can't find the specified class.

4.2.2.12 #define E_CLASS_NOT_SUPPORTED (-83)

Attempt to access member of wrong class type.

4.2.2.13 #define E_COMMUNICATIONS_ERROR (-34)

Framework communications failure.

4.2.2.14 #define E_CONSTRUCTOR_ARGUMENT_COUNT (-4)

A constructor was called with the wrong number of arguments.

4.2.2.15 #define E_DUPLICATE_ID (-68)

Attempt to set duplicate ID with SetID.

4.2.2.16 #define E_EXCEPTION (-74)

CFramework exception (null pointer?).

4.2.2.17 #define E_FILENAMELENGTH (-41)

Filename length $>$ 23.

4.2.2.18 #define E_HEAP_INDEX_RANGE (-1)

Attempt to reference a non-existent heap.

4.2.2.19 #define E_ID_OUT_OF_RANGE (-69)

Attempt to use ID outside 1..32767 with SetID.

4.2.2.20 #define E_INCOMPATIBLE_BLOCK_SIZES (-29)

For containers, the contained module required sizes don't match.

4.2.2.21 #define E_INPUT_OUTPUT_COUNT (-16)

Module with fixed inputs/outputs has wrong count.

4.2.2.22 #define E_INPUTS_MUST_BE_SAME_PINTYPE (-11)

Many modules require that all inputs be of the same pintype.

4.2.2.23 #define E_INPUTS_MUST_MATCH_CORRESPONDING_OUTPUTS (-14)

Many modules require the input and output types to match in pairs.

4.2.2.24 #define E_INPUTS_MUST_MATCH_OUTPUTS (-10)

Many modules require the number of inputs and outputs to be the same.

4.2.2.25 #define E_INVALID_FILE (-95)

The file content is invalid.

4.2.2.26 #define E_IOPIN_IN_USE (-21)

The I/O pin given to ClassXXWire_Constructor() is already in use.

4.2.2.27 #define E_IOPIN_TOO_MANY (-88)

Too many wires bound to pin.

4.2.2.28 #define E_LINKEDLIST_CORRUPT (-94)

Module instance linked list is corrupted.

4.2.2.29 #define E_MALLOC_SIZE_TOO_BIG (-2)

Attempt to allocate more storage with awe_fwMalloc() than exists.

4.2.2.30 #define E_MODULE_ALREADY_OWNED (-7)

Attempt to bind a module instance to a layout that is already owned by a layout.

4.2.2.31 #define E_MODULE_NOT_INITIALIZED (-96)

The module was not initialized.

4.2.2.32 #define E_MODULE_NOT_WIRE (-9)

A wire input given to a module constructor is not a wire.

4.2.2.33 #define E_MSG_TIMEOUT (-78)

Message timed out (C6 only).

4.2.2.34 #define E_MUST_HAVE_INPUTS (-12)

Many modules require at leat one input.

4.2.2.35 #define E_MUST_HAVE_ONE_OUTPUT (-27)

For modules with a hardwired single output (like RMS).

4.2.2.36 #define E_MUST_HAVE_OUTPUTS (-13)

Many modules require at leat one output.

4.2.2.37 #define E_NO_CORE (-87)

The specified core does not exist.

4.2.2.38 #define E_NO_EVENT_HANDLER (-31)

There is no event handler for this module.

4.2.2.39 #define E_NO_FRAMEWORK (-86)

The framework does not exist.

4.2.2.40 #define E_NO_HEAP_MEMORY (-66)

Attempt to create heap failed.

4.2.2.41 #define E_NO_LAYOUTS (-26)

Master pump function found no layouts to pump.

4.2.2.42 #define E_NO_MORE_IOPINS (-25)

awe_fwGetFirstIO() or awe_fwGetNextIO() reached end of object chain.

4.2.2.43 #define E_NO_MORE_OBJECTS (-18)

awe_fwGetFirstObject() or awe_fwGetNextObject() reached end of object chain.

4.2.2.44 #define E_NOSUCHFILE (-44)

The specified file does not exist.

4.2.2.45 #define E_NOT_CREATED (-91)

AWE instance not created.

4.2.2.46 #define E_NOT_IMPLEMENTED_IN_RS232 (-38)

Attempt to pump for RS232.

4.2.2.47 #define E_NOT_INPUT_PIN (-20)

Object pointer given to ClassInputWire_Constructor() is not an input pin.

4.2.2.48 #define E_NOT_MODULE (-15)

The module given is not a module.

4.2.2.49 #define E_NOT_OBJECT (-81)

Not a valid object.

4.2.2.50 #define E_NOT_OBJECT_POINTER (-19)

Object pointer given to awe_fwGetNextObject() does not point to a module instance.

4.2.2.51 #define E_NOT_OUTPUT_PIN (-24)

Object pointer given to ClassOutpuWire_Constructor() is not an output pin.

4.2.2.52 #define E_NOT_V5 (-85)

Target is not a V5 target - legacy.

4.2.2.53 #define E_OUTPUT_MUST_BE_SINGLE_VALUE (-28)

For modules that need an output with 1 single output value (like RMS).

4.2.2.54 #define E_PARAMETER_ERROR (-17)

Module arguments have an error.

4.2.2.55 #define E_PIN_ID_NOT_FOUND (-80)

I/O pin not found.

4.2.2.56 #define E_PIN_SIZES_NOT_COMPATIBLE (-23)

The wire and I/O pin sizes are not compatible.

4.2.2.57 #define E_PIN_TYPES_NOT_COMPATIBLE (-22)

The wire and I/O pin type are not compatible.

4.2.2.58 #define E_PUMP_OVERRUN (-84)

Attempt to pump when awe_fwTick() was not called.

4.2.2.59 #define E_READ_ONLY (-70)

Attempt to modify read-only object header.

4.2.2.60 #define E_SCRATCH_ALLOC_SIZE_TOO_BIG (-3)

Attempt to allocate more storage with awe_fwMallocScratch() than exists.

4.2.2.61 #define E_WIRE_ALREADY_BOUND (-89)

Attempt to bind a wire already bound.

4.2.2.62 #define E_WIRE_INDEX_RANGE (-30)

A write index in a vector is out of range.

4.2.2.63 #define E_WIRES_NOT_SPECIFIED (-90)

Required wires not specified.

5 Class Documentation

5.1 _Sample Union Reference

Union of all possible sample types.

`#include <awelib.h>`

Public Attributes

- INT32 **iVal**
- UINT32 **uiVal**
- float **fVal**

5.1.1 Detailed Description

Union of all possible sample types.

5.2 CAWELib Class Reference

Interface for the AudioWeaver library.

`#include <awelib.h>`

Public Member Functions

- virtual ~CAWELib ()

  Destructor.
- virtual double AverageCycles () const =0

  Get the average cycles used by audio.
- virtual UINT64 PumpCount () const =0

  Get the pump count.
- virtual void ClearMeasure ()=0

  Clear measure of average cycles.
- virtual const char * GetLibraryVersion () const =0

  Get the library version.
- virtual void SetLayoutAddresses (UINT32 inId, UINT32 outId, UINT32 layoutId=0)=0

  Specify addresses in layout.
- virtual bool IsCreated () const =0

  test if the AWE instance has been created
- virtual int Create (const char *name, float cpu_clock_speed, float cpu_cycle_speed)=0

  Create the AWE instance.
- virtual void Destroy ()=0

  Destroy the AWE instance.
- virtual int PumpAudio (const void *in_samples, void *out_samples, UINT32 in_sample_count, UINT32 out_sample_count)=0

  Write samples, pump, and read samples.
- virtual int PinProps (UINT32 &in_samps, UINT32 &in_chans, UINT32 &in_complex, UINT32 &in_ID, UINT32 &out_samps, UINT32 &out_chans, UINT32 &out_complex, UINT32 &out_ID)=0

  Query the wire properties.
- virtual Sample FetchValue (UINT32 address, int *retVal)=0

  Fetch a 32-bit value from the given address.

- virtual int SetValue (UINT32 address, Sample val)=0

  Set a 32-bit value to the given address.
- virtual int FetchValues (UINT32 address, UINT32 argCount, Sample *args, UINT32 offset=0)=0

  Fetch array values.
- virtual int SetValues (UINT32 address, UINT32 argCount, const Sample *args, UINT32 offset=0)=0

  Set array values.
- virtual int LoadAwbFile (const char *filename, UINT32 *pPos=0)=0

  Load an AWB file into AWE.
- virtual int LoadAwbMem (UINT32 *buffer, UINT32 len, UINT32 *pPos=0)=0

  Load an AWB file from memory into AWE.
- virtual int SendCommand (UINT32 *cmdBuffer, UINT32 cmdBufferLen)=0

  Send a binary command to AWE.
- virtual UINT32 GetCycles ()=0

  Get the cycle counter.
- virtual bool IsStarted ()=0

  Test if audio is running.
- virtual int PinPropsEx (UINT32 &in_samps, UINT32 &in_chans, UINT32 &in_complex, UINT32 &in_ID, UINT32 &out_samps, UINT32 &out_chans, UINT32 &out_complex, UINT32 &out_ID, UINT32 *pSrate=0)=0

  Query the wire properties with optional sample rate.
- virtual int CreateEx (const char *name, float cpu_clock_speed, float cpu_cycle_speed, bool profile=false, UINT32 blocksize=0, UINT32 bufferlen=0, UINT32 inchans=0, UINT32 outchans=0)=0

  Create the AWE instance with profile control.
- virtual void SetStartStopCallbacks (void *param, void(*start)(void *), void(*stop)(void *))=0

  Set callback handlers for start and stop audio events.
- virtual void SetSearchPath (const char *searchPath)=0

  Set the file search path.
- virtual int LoadAwbFileDSPCEncrypted (const char *filename, UINT32 *pPos=0)=0

  Load a DSPC encrypted AWB file into AWE.
- virtual int LoadAwbFileUnencrypted (const char *filename, UINT32 *pPos=0)=0

  Load an unencrypted AWB file into AWE.
- virtual int LoadAwbMemDSPCEncrypted (UINT32 *buffer, UINT32 len, UINT32 *pPos=0)=0

  Load a DSPC encrypted AWB file from memory into AWE.
- virtual int LoadAwbMemUnencrypted (UINT32 *buffer, UINT32 len, UINT32 *pPos=0)=0

  Load an unencrypted AWB file from memory into AWE.
- virtual int CreateEx2 (const char *name, float cpu_clock_speed, float cpu_cycle_speed, bool profile=false, UINT32 blocksize=0, UINT32 bufferlen=0, UINT32 inchans=0, UINT32 outchans=0, UINT32 masterHeap_size=0, UINT32 fastHeapB_size=0, UINT32 slowHeap_size=0)=0

  Create the AWE instance with profile control and configurable heaps.

5.2.1 Detailed Description

Interface for the AudioWeaver library.

This is an abstract interface to the AWE library. You construct an instance by calling AWELibraryFactory(). The constructed instance is not usable until you call Create() which creates an empty AWE, and can't be used to process audio until you call LoadAwbFile() to instantiate audio modules.

After loading the AWB file, you must call PinProps() to get the layout properties, and pass the wire IDs to SetLayoutAddresses(). You can then call PumpAudio() to process blocks of samples.

The AWB file you load is a binary representation of a Designer layout you create. The block size and channel counts for input and output are determined by the layout design, which is why you must call PinProps() to query those values.

See provided example code for details of how to do this.

5.2.2 Member Function Documentation

5.2.2.1 virtual double CAWELib::AverageCycles ( ) const [pure virtual]

Get the average cycles used by audio.

Returns

the average cycles

The return value is in units of the sampling clock used by your CPU. This should be the same as the value you specified when you called Create(). The value is determined by hardware. See your processor data sheet.

5.2.2.2 virtual void CAWELib::ClearMeasure ( ) [pure virtual]

Clear measure of average cycles.

5.2.2.3 virtual int CAWELib::Create (const char * *name*, float *cpu_clock_speed*, float *cpu_cycle_speed*) [pure virtual]

Create the AWE instance.

Parameters

in	*name*	instance name (max 8 chars)
in	*cpu_clock_speed*	CPU clock speed in Hz
in	*cpu_cycle_speed*	sample clock speed in Hz

Returns

0 or error code

Creates the AWE instance. All other APIs will fail until you call this. You must specify a name for this instance (max 8 chars), the CPU clock frequency and sample clock frequency (usually the same).

Cycle speed is the clock used for cycle counting. On a number of CPUs this is not the same as the CPU clock. The value is used for Matlab profiling to compute MIPs. See your processor data sheet.

On many Linux systems and on Windows the profile value should be 10000000 (10MHz).

Possible error returns: E_ALREADY_CREATED - Create() has already been called. E_ARGUMENT_ERROR - invalid name.

5.2.2.4 virtual int CAWELib::CreateEx (const char * *name*, float *cpu_clock_speed*, float *cpu_cycle_speed*, bool *profile* = false, UINT32 *blocksize* = 0, UINT32 *bufferlen* = 0, UINT32 *inchans* = 0, UINT32 *outchans* = 0) [pure virtual]

Create the AWE instance with profile control.

Parameters

in	*name*	instance name (max 8 chars)
in	*cpu_clock_speed*	CPU clock speed in Hz
in	*cpu_cycle_speed*	sample clock speed in Hz

in	*profile*	default false, pass true to enable profiling
in	*blocksize*	default 0, pass blocksize target info must report
in	*bufferlen*	default 0, pass command buffer length target info must report
in	*inchans*	default 0, pass number of input channels target info must report
in	*outchans*	default 0, pass number of output channels target info must report

Returns

0 or error code

Creates the AWE instance. All other APIs will fail until you call this. You must specify a name for this instance (max 8 chars), the CPU clock frequency and sample clock frquency (usually the same).

Cycle speed is the clock used for cycle counting. On a number of CPUs this is not the same as the CPU clock. The value is used for Matlab profiling to compute MIPs. See your processor data sheet.

On many Linux systems and on Windows the cycle value should be 10000000 (10MHz).

Possible error returns: E_ALREADY_CREATED - Create() has already been called. E_ARGUMENT_ERROR - invalid name.

5.2.2.5 virtual int CAWELib::CreateEx2 (const char * *name*, float *cpu_clock_speed*, float *cpu_cycle_speed*, bool *profile* = `false`, UINT32 *blocksize* = `0`, UINT32 *bufferlen* = `0`, UINT32 *inchans* = `0`, UINT32 *outchans* = `0`, UINT32 *masterHeap_size* = `0`, UINT32 *fastHeapB_size* = `0`, UINT32 *slowHeap_size* = `0`) [pure virtual]

Create the AWE instance with profile control and configurable heaps.

Parameters

in	*name*	instance name (max 8 chars)
in	*cpu_clock_speed*	CPU clock speed in Hz
in	*cpu_cycle_speed*	sample clock speed in Hz
in	*profile*	default false, pass true to enable profiling
in	*blocksize*	default 0, pass blocksize target info must report
in	*bufferlen*	default 0, pass command buffer length target info must report
in	*inchans*	default 0, pass number of input channels target info must report
in	*outchans*	default 0, pass number of output channels target info must report
in	*masterHeap_size*	default 0, pass in master heap size that will be allocated
in	*fastHeapB_size*	default 0, pass in fast heap B size that will be allocated
in	*slowHeap_size*	default 0, pass in slow heap size that will be allocated

Returns

0 or error code

Creates the AWE instance. All other APIs will fail until you call this. You must specify a name for this instance (max 8 chars), the CPU clock frequency and sample clock frquency (usually the same). This updated CreateEx2 function allows heap sizes to be passed in (as opposed to the heap sizes built into the library).

Cycle speed is the clock used for cycle counting. On a number of CPUs this is not the same as the CPU clock. The value is used for Matlab profiling to compute MIPs. See your processor data sheet.

On many Linux systems and on Windows the cycle value should be 10000000 (10MHz).

Possible error returns: E_ALREADY_CREATED - Create() has already been called. E_ARGUMENT_ERROR - invalid name.

5.2.2.6 virtual void CAWELib::Destroy ( ) [pure virtual]

Destroy the AWE instance.

This destroys the AWE instance. You can later call Create() to create a new instance.

5.2.2.7 virtual Sample CAWELib::FetchValue (UINT32 *address*, int * *retVal*) [pure virtual]

Fetch a 32-bit value from the given address.

Parameters

in	*address*	Address to fetch value from.
out	*retVal*	Error reason code.

Returns

Fetched value.

AWE stores array values and scalar values differently use this when reading scalar values.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

5.2.2.8 virtual int CAWELib::FetchValues (UINT32 *address*, UINT32 *argCount*, Sample * *args*, UINT32 *offset* = 0) [pure virtual]

Fetch array values.

Parameters

in	*address*	base address
in	*argCount*	count of values
out	*args*	the output values
in	*offset*	offset from base default 0

Returns

0 or error code

AWE stores array values and scalar values differently use this when reading array values even if only one element.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

5.2.2.9 virtual UINT32 CAWELib::GetCycles ( ) [pure virtual]

Get the cycle counter.

Returns

the cycles

The return value is in units of the cycle counter frequency.

5.2.2.10 virtual const char* CAWELib::GetLibraryVersion ( ) const [pure virtual]

Get the library version.

Returns

the library version string

5.2.2.11 virtual bool CAWELib::IsCreated ( ) const [pure virtual]

test if the AWE instance has been created

Returns

true if it has

5.2.2.12 virtual bool CAWELib::IsStarted ( ) [pure virtual]

Test if audio is running.

Returns

true if it is

5.2.2.13 virtual int CAWELib::LoadAwbFile (const char * *filename*, UINT32 * *pPos* = 0) [pure virtual]

Load an AWB file into AWE.

Parameters

in	*filename*	file to load
out	*pPos*	if non-NULL, return failing word offset

Returns

0 or error code

Loads a layout file in AWB format. After loading a layout, you should call PinProps() to find out the layout input and output details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_ARGUMENT_ERROR - filename is not valid. Others - see Errors.h.

On error, if pPos is given, the word offset of the failing command is returned.

5.2.2.14 virtual int CAWELib::LoadAwbFileDSPCEncrypted (const char * *filename*, UINT32 * *pPos* = 0) [pure virtual]

Load a DSPC encrypted AWB file into AWE.

Parameters

in	*filename*	file to load
out	*pPos*	if non-NULL, return failing word offset

Returns

0 or error code

Loads a layout file in AWB format. After loading a layout, you should call PinProps() to find out the layout input and output details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_ARGUMENT_ERROR - filename is not valid. Others - see Errors.h.

On error, if pPos is given, the word offset of the failing command is returned.

5.2.2.15 virtual int CAWELib::LoadAwbFileUnencrypted (const char * *filename*, UINT32 * *pPos* = 0) [pure virtual]

Load an unencrypted AWB file into AWE.

Parameters

in	*filename*	file to load
out	*pPos*	if non-NULL, return failing word offset

Returns

0 or error code

Loads a layout file in AWB format. After loading a layout, you should call PinProps() to find out the layout input and output details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_ARGUMENT_ERROR - filename is not valid. Others - see Errors.h.

On error, if pPos is given, the word offset of the failing command is returned.

5.2.2.16 virtual int CAWELib::LoadAwbMem (UINT32 * *buffer*, UINT32 *len*, UINT32 * *pPos* = 0) [pure virtual]

Load an AWB file from memory into AWE.

Parameters

in	*buffer*	buffer containing words to load
in	*len*	number of words in buffer
out	*pPos*	if non-NULL, return failing word offset

Returns

0 or error code

Loads a layout file in AWB format that has been read into an array of words as though by using fread(). Afterloading a layout, you should call PinProps() to find out the layout input and output details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_ARGUMENT_ERROR - buffer or len is not valid. Others - see Errors.h.

On error, if pPos is given, the word offset of the failing command is returned.

5.2.2.17 virtual int CAWELib::LoadAwbMemDSPCEncrypted (UINT32 * *buffer*, UINT32 *len*, UINT32 * *pPos* = 0) [pure virtual]

Load a DSPC encrypted AWB file from memory into AWE.

Parameters

in	*buffer*	buffer containing words to load
in	*len*	number of words in buffer
out	*pPos*	if non-NULL, return failing word offset

Returns

0 or error code

Loads a layout file in AWB format that has been read into an array of words as though by using fread(). Afterloading a layout, you should call PinProps() to find out the layout input and output details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_ARGUMENT_ERROR - buffer or len is not valid. Others - see Errors.h.

On error, if pPos is given, the word offset of the failing command is returned.

5.2.2.18 virtual int CAWELib::LoadAwbMemUnencrypted (UINT32 * *buffer*, UINT32 *len*, UINT32 * *pPos* = 0) [pure virtual]

Load an unencrypted AWB file from memory into AWE.

Parameters

in	*buffer*	buffer containing words to load
in	*len*	number of words in buffer
out	*pPos*	if non-NULL, return failing word offset

Returns

0 or error code

Loads a layout file in AWB format that has been read into an array of words as though by using fread(). Afterloading a layout, you should call PinProps() to find out the layout input and output details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_ARGUMENT_ERROR - buffer or len is not valid. Others - see Errors.h.

On error, if pPos is given, the word offset of the failing command is returned.

5.2.2.19 virtual int CAWELib::PinProps (UINT32 & *in_samps*, UINT32 & *in_chans*, UINT32 & *in_complex*, UINT32 & *in_ID*, UINT32 & *out_samps*, UINT32 & *out_chans*, UINT32 & *out_complex*, UINT32 & *out_ID*) [pure virtual]

Query the wire properties.

Parameters

out	*in_samps*	input wire samples
out	*in_chans*	input wire channels
out	*in_complex*	input wire complex
out	*in_ID*	ID of input
out	*out_samps*	output wire samples
out	*out_chans*	output wire channels
out	*out_complex*	output wire complex
out	*out_ID*	ID of output

Returns

0, 1, or error code

Queries the layout for the input and output details. Wires are almost never complex, but if they are there are two values (real, imag) per logical sample. Return the block size, channels, complex, and ID of the the wires. If there is no layout, returns 0, otherwise returns 1 and these details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

5.2.2.20 virtual int CAWELib::PinPropsEx (UINT32 & *in_samps*, UINT32 & *in_chans*, UINT32 & *in_complex*, UINT32 & *in_ID*, UINT32 & *out_samps*, UINT32 & *out_chans*, UINT32 & *out_complex*, UINT32 & *out_ID*, UINT32 * *pSrate* = 0) [pure virtual]

Query the wire properties with optional sample rate.

Parameters

out	*in_samps*	input wire samples
out	*in_chans*	input wire channels
out	*in_complex*	input wire complex
out	*in_ID*	ID of input
out	*out_samps*	output wire samples

out	*out_chans*	output wire channels
out	*out_complex*	output wire complex
out	*out_ID*	ID of output
out	*pSrate*	optional layout sample rate

Returns

0, 1, or error code

Queries the layout for the input and output details. Wires are almost never complex, but if they are there are two values (real, imag) per logical sample. Return the block size, channels, complex, and ID of the the wires. If there is no layout, returns 0, otherwise returns 1 and these details.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

5.2.2.21 virtual int CAWELib::PumpAudio (const void * *in_samples*, void * *out_samples*, UINT32 *in_sample_count*, UINT32 *out_sample_count*) [pure virtual]

Write samples, pump, and read samples.

Parameters

in	*in_samples*	samples to send to AWE
out	*out_samples*	where to write reply samples
in	*in_sample_count*	number of input samples
in	*out_sample_count*	number of output samples

Returns

0 or error code

Processes the input samples through the layout to the output samples and keeps track of the cycles used by audio processing. The wire IDs must have been specified first, or this call will fail. The counts must be block size * channels.

If audio has not been started by the layout, it is safe to call this but it will not process samples, just clear the output buffer and return E_NO_LAYOUTS or E_AUDIO_NOT_STARTED.

Possible error returns: E_NOT_CREATED - Create() must be called before use. E_WIRES_NOT_SPECIFIED - wire IDs were not set by SetLayoutAddresses(). E_NO_LAYOUTS - no layout, use LoadAwbFile(). E_AUDIO_NOT_STARTED - audio has not yet started, start it with SendCommand() or use LoadAwbFile(). Others - see Errors.h.

5.2.2.22 virtual UINT64 CAWELib::PumpCount ( ) const [pure virtual]

Get the pump count.

Returns

the pump count

5.2.2.23 virtual int CAWELib::SendCommand (UINT32 * *cmdBuffer*, UINT32 *cmdBufferLen*) [pure virtual]

Send a binary command to AWE.

Parameters

	[in	out] cmdBuffer buffer containing command
in	*cmdBufferLen*	buffer size in words

Returns

0 or error code

Warning: replies can be much larger than commands. The buffer should be sized at MAX_COMMAND_BUFFER-_LEN to handle the worst case. On entry, the buffer must contain a binary AWE command packet. On return, the buffer contains the reply packet.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

5.2.2.24 virtual void CAWELib::SetLayoutAddresses (UINT32 *inId*, UINT32 *outId*, UINT32 *layoutId* = 0) [pure virtual]

Specify addresses in layout.

Parameters

in	*inId*	ID of input wire
in	*outId*	ID of output wire
in	*layoutId*	ID of layout default 0

You must set the wire IDs for input and output before you can pump. A layout ID of 0 means use the default layout.

5.2.2.25 virtual void CAWELib::SetSearchPath (const char * *searchPath*) [pure virtual]

Set the file search path.

Parameters

searchPath	[in] vertical bar delimited path

Some modules that read files use a search path to locate them. The items in this path must be '|' delimited. If not called, a path of '.' is assumed, meaning the working directory.

5.2.2.26 virtual void CAWELib::SetStartStopCallbacks (void * *param*, void(*)(void *) *start*, void(*)(void *) *stop*) [pure virtual]

Set callback handlers for start and stop audio events.

Parameters

in	*param*	parameter to pass to callbacks
in	*start*	start event handler
in	*stop*	stop event handler

You can disable callbacks again by passing NULLs.

The callbacks are only called when the audio state changes.

5.2.2.27 virtual int CAWELib::SetValue (UINT32 *address*, Sample *val*) [pure virtual]

Set a 32-bit value to the given address.

Parameters

in	*address*	Address to set value to
in	*val*	value to set

Returns

0 or error code

AWE stores array values and scalar values differently use this when writing scalar values.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

5.2.2.28 virtual int CAWELib::SetValues (UINT32 *address*, UINT32 *argCount*, const Sample * *args*, UINT32 *offset* = 0) [pure virtual]

Set array values.

Parameters

in	*address*	base address
in	*argCount*	count of values
in	*args*	the argument values
in	*offset*	offset from base default 0

Returns

0 or error code

AWE stores array values and scalar values differently use this when writing array values even if only one element.

Possible error returns: E_NOT_CREATED - Create() must be called before use. Others - see Errors.h.

Index