HEAAC Decoder

XDM API

Document Name	IA-HEAAC-DEC-XDM-API
Version	2.0
Date	July 24, 2008

Ittiam Systems Confidential

Ittiam Systems (P) Ltd,
The Consulate, 1 Richmond Road,
Bangalore 560 025, India

Notice

Ittiam Systems reserves the right to make changes to its products or discontinue any of its products or offerings without notice.

Ittiam warrants the performance of its products to the specifications applicable at the time of sale in accordance with Ittiam's standard warranty.

Revision History

Version	Date	Changes
1.0	July 31, 2006	Original
1.1	January 15, 2007	Updated for multichannel structure contents
1.2	January 31, 2007	Added valid values for extended parameters
1.3	February 14, 2007	Updated for XDM 0.9
1.4	August 1, 2007	Corrected the MPEG expansion
1.5	December 7, 2007	Added new set parameters, get parameters and error codes
1.6	March 28, 2008	Updated outArgs with output frame length information
2.0	July 24, 2008	Updated to xDM 1.0

Contents

Figures

Tables

1. Introduction

1.1 Motivation

AAC is a popular audio coding technique recommended by MPEG committee. The codec handles audio signals sampled in the range of 8 kHz to 96 kHz. It operates on a frame of 1024 samples. The bit-rates supported are in the range of 8 kbps to 576 kbps per channel.

SBR and PS are the tools used in combination with the AAC general audio codec. Refer **[1]** and **[2]** for details. This results in Enhanced aacPlus (also known as HEAAC v2). Usage of Enhanced aacPlus results in significant increase of coding gain. In SBR, the high-band, i.e. the high frequency part of the spectrum is replicated using the low-band. In PS, channel redundancy is exploited and parameters are sent to recreate a stereo output from a down-mixed channel. By using these tools the bit-rate required is much lower than the conventional AAC coding bit-rates. Or in other terms you achieve better quality at lower bit-rates.

This document describes the **A**pplication **P**rogram **I**nterface for the Enhanced aacPlus Decoder. It also addresses the knowledge requirements of developers to integrate different components of their system with Ittiam Enhanced aacPlus Decoder software solution.

1.2 Scope

The document assumes that the reader has sufficient information about XDAIS and XDM APIs as defined by TI. The version of header files used for reference is “Version 1.0”. This doc details all the elements in interface data structures for this component. Along with the component specific information, generic information is also elaborated in certain cases where information provided by TI is ambiguous or incomplete.

The document will provide knowledge to developers in terms of the following:

- Interface functions, commands and data structures (**Chapter 2**)
  - This chapter gives a complete overview of the API, commands, and interface structures.
- API Integration (**Chapter 3**)
  - It contains the integration flow-chart of the decoder.
  - Overview of error codes.

1.3 Glossary

API	Application Program Interface (Interface through which an application talks to functional blocks)
MPEG	Moving Picture Experts Group
SBR	Spectral Band Replication
PS	Parametric Stereo
XDAIS	eXpress DSP Algorithm Interoperability Standard
XDM	XDAIS Digital Media eXpress DSP Algorithm Interoperability (Digital Media) Standard

2. Interfaces

In this section the list of function interfaces available for this component will be described.

2.1 Generic notes

Here are few generic notes about all interface data structures used for this component.

1. "`size`" parameter is checked inside the algorithm to see if the passed structure is same as expected. The API handles both scenarios of sending in only the standard structures and sending in an extended structure. Application in all cases is expected to fill this element properly for all structures before communicating with the component.
2. ImplementationId field in the IMOD function vector table is used to check if the right object handle is passed for processing or not. This feature will be useful to resolve some application level bugs. If wrong object handle is passed, algorithm simply returns the call and returns error if return is allowed in that call.
3. Default parameters in `ENHAACPDECODER_ITTIAM_PARAMS` can be used to get basic set of configuration parameters. Application needs to update only required fields.
4. Original documentation of standard interfaces can be found in **`[2] & [3]`**.

2.2 API of Enhanced aacPlus Decoder

2.2.1 Functional Interface

ITTIAM_ENHAACPDEC_Fxns

Description: This structure contains all the functional interfaces available for the component.

Syntax:

```
typedef struct ITTIAM_ENHAACPDEC_Fxns{
  IAUDDEC1_Fxns s_iauddec_fxns
  {
    IALG_Fxns ialg
    {
      Void *implementationId;
      Void (*algActivate)(IALG_Handle);
```

```
    Int (*algAlloc)(const IALG_Params *,
      struct IALG_Fxns **, IALG_MemRec *);
    Int (*algControl)(IALG_Handle, IALG_Cmd,
      IALG_Status *);
    Void (*algDeactivate)(IALG_Handle);
    Int (*algFree)(IALG_Handle, IALG_MemRec *);
    Int (*algInit)(IALG_Handle,
      const IALG_MemRec *, IALG_Handle,
      const IALG_Params *);
    Void (*algMoved)(IALG_Handle,
      const IALG_MemRec *, IALG_Handle,
      const IALG_Params *);
    Int (*algNumAlloc)(Void);
  }
  XDAS_Int32 (*process)(
    IAUDDEC1_Handle handle,
    XDM1_BufDesc *inBufs,
    XDM1_BufDesc *outBufs,
    IAUDDEC1_InArgs *inArgs,
    IAUDDEC1_OutArgs *outArgs)
  XDAS_Int32 (*control)(
    IAUDDEC1_Handle handle,
    IAUDDEC1_Cmd id,
    IAUDDEC1_DynamicParams *params,
    IAUDDEC1_Status *status);
 }
} ITTIAM_ENHAACPDEC_Fxns;
```

Elements	Functions	Operations
	s_auddec_fxns.ialg.	
	implementationId	Unique pointer that identifies the module implementing this interface
	algActivate	Notification to the algorithm that its memory is "active" and algorithm processing methods may be called. NOT IMPLEMENTED
	algAlloc	Apps call this to query the algorithm about its memory requirements.
	algControl	Algorithm specific control and status. NOT IMPLEMENTED
	algDeactivate	Save all persistent data to non-scratch memory. NOT IMPLEMENTED
	algFree	Apps call this to allow the algorithm to initialize memory requested via algAlloc().
	algInit	Initialize an algorithm's instance object.
	algMoved	Notify algorithm instance that instance memory has been relocated

algNumAlloc	Returns number of memory allocation requests required
s_auddec_fxns.process	Main process call for the Enhanced aacPlus Decoder operation. It takes in an input bit-stream and generates output samples. In case of fatal errors it returns IAUDDEC1_EFAIL or IAUDDEC1_EUNSUPPORTED. Codec specific error code got from OutArgs->s_iauddec_outargs.extendedError & OutArgs->i_ittiam_err_code. For details refer Section 3.3 on error codes.

s_auddec_fxns.control

Commands	Operations
XDM_GETSTATUS	Query algorithm to fill status structure. For description of elements & their valid values refer section 2.2.4.3
XDM_SETPARAMS	Set run time dynamic parameters. For description of elements & their valid values refer section 2.2.4.2
XDM_RESET	Reset the algorithm. All fields in the internal data structures are reset and all internal buffers are flushed. Ittiam Implementation: The decoder returns to the state it was just after algInit.
XDM_SETDEFAULT	Restore the algorithm's internal state to its original, default values Ittiam Implementation: In particular the dynamic parameters stored in the internal state are restored to original default values.
XDM_FLUSH	Handle end of stream conditions. This command forces the algorithm to output data without additional input. The recommended sequence is to call the control() function (with XDM_FLUSH) followed by repeated calls to the process() function until it returns an error. Ittiam Implementation: A flag to indicate input over is set and from the second process call after setting this flag IAUDDEC1_EFAIL is returned.
XDM_GETBUFINFO	Query the algorithm instance regarding its properties of input and output buffers. Only the bufInfo element of the status structure is filled.
XDM_GETVERSION	Query the algorithm's version. The result will be returned in the data field of the status structure

Table 2.1 API Functions

2.2.2 Input/output Format

2.2.2.1 Input Format

It takes compressed data in byte format.

2.2.2.2 Output Format

The generated output is 16 bits PCM samples.

2.2.3 Default Parameters

ENHAACPDECODER_ITTIAM_PARAMS

Description: This contains the default initialization parameters for the component.

Syntax:

```
typedef struct ITTIAM_ENHAACPDEC_Params {
  IAUDDEC1_Params s_iauddec_params;
  {
    XDAS_Int32 size
    XDAS_Int32 outputPCMWidth;
    XDAS_Int32 pcmFormat;
    XDAS_Int32 dataEndianness;
  }
  /* ITTIAM's extensions */
  XDAS_Int32      i_max_channels;
  XDAS_Int32      i_flag_08khz_out;
  XDAS_Int32      i_flag_16khz_out;
  XDAS_Int32      i_interleave;
  XDAS_Int32      i_mp4_flag;
  XDAS_Int32      i_disable_sync;
  XDAS_Int32      i_auto_sbr_upsample;
  XDAS_Int32      i_sampfreq;
  XDAS_Int32      i_coupling_channel;
} ITTIAM_ENHAACPDEC_Params;
```

Parameters:

Names	Values
size	sizeof(ITTIAM_ENHAACPDEC_Params)
outputPCMWidth	16
pcmFormat	IAUDIO_INTERLEAVED
dataEndianness	XDM_LE_16
i_max_channels	2 (for stereo libraries), 6 (for multi-channel libraries)
i_flag_08khz_out	0
i_flag_16khz_out	0

i_interleave	1
i_mp4_flag	0
i_disable_sync	0
i_auto_sbr_upsample	1
i_sampfreq	0
i_coupling_channel	0

Usage: Application gets a suitable set of configuration parameters and may update only a few required elements.

Table 2.2 Default parameters

2.2.4 Data structures

2.2.4.1 ITTIAM_ENHAACPDEC_Params

ITTIAM_ENHAACPDEC_Params

Description: This data structure is used by the application to convey configuration parameters to the algorithm.

Syntax:

```
typedef struct ITTIAM_ENHAACPDEC_Params {
  IAUDDEC1_Params s_iauddec_params;
  {
    XDAS_Int32 size
    XDAS_Int32 outputPCMWidth;
    XDAS_Int32 pcmFormat;
    XDAS_Int32 dataEndianness;
  }
  /* ITTIAM's extensions */
  XDAS_Int32      i_max_channels;
  XDAS_Int32      i_flag_08khz_out;
  XDAS_Int32      i_flag_16khz_out;
  XDAS_Int32      i_interleave;
  XDAS_Int32      i_mp4_flag;
  XDAS_Int32      i_disable_sync;
  XDAS_Int32      i_auto_sbr_upsample;
  XDAS_Int32      i_sampfreq;
  XDAS_Int32      i_coupling_channel;
} ITTIAM_ENHAACPDEC_Params;
```

Parameters:

size

Size of this structure in bytes.

outputPCMWidth

Number of bits per output PCM Sample. Only 16 bits per sample is supported. If any other value is sent an IAUDDEC1_EUNSUPPORTED error is returned.

pcmFormat

Output PCM Format Block/Interleaved. Only IAUDIO_INTERLEAVED is supported. If any other

value is sent an IAUDDEC1_EUNSUPPORTED error is returned.

dataEndianness

Endianness of output data. Only XDM_LE_16 is supported. If any other value is sent an IAUDDEC1_EUNSUPPORTED error is returned.

i_max_channels

Max number of channels (including LFE channels) which the decoder needs to be prepared to decode.

For multichannel build a range of 2 to 8 is allowed. If values beyond this range are sent it will be limited to this range and used.

For stereo builds this value is not used and the decoder will always be able to decode up to 2 channels.

This element to some extent decides the amount of persistent and scratch memory to be allocated.

i_flag_08khz_out

Resample the output to 8 kHz. Valid values are 0 or 1. If any other value is sent an IAUDDEC1_EFAIL error is returned. The usage of this flag is not implemented in any of the default builds.

i_flag_16khz_out

Resample the output to 16 kHz. Valid values are 0 or 1. If any other value is sent an IAUDDEC1_EFAIL error is returned. The usage of this flag is not implemented in any of the default builds.

i_interleave

Interleave mono output to stereo. Valid values are 0 or 1. If any other value is sent, default value is taken as 1.

i_mp4_flag

To indicate if an AudioSpecifConfig header (the audio header present in MP4 files) will be provided or not. Valid values are 0 or 1. If any other value is sent an IAUDDEC1_EFAIL error is returned.

NOTE: When this flag is enabled then ONLY AudioSpecificConfig header data should be provided in the first call to the decoder. From the next call onwards audio data should be provided.

i_disable_sync

To indicate if ADIF/ADTS synchronization is to be disabled. In which case the header is expected to be at the beginning of the input buffer. Valid values are 0 or 1. If any other value is sent an IAUDDEC1_EFAIL error is returned.

i_auto_sbr_upsample

To indicate if automatic SBR up sampling has to enabled or disabled in cases of stream changing

from SBR present to SBR not present. Valid values are 0 or 1. If any other value is sent an IAUDDEC1_EFAIL error is returned.

i_sampfreq

To indicate if a RAW stream is being provided and to indicate the core AAC sample rate for that bit-stream. Note that the core AAC sample rate might be different from the output sample rate for files with SBR. Valid values are '0' or valid sampling rates in range of 8000 to 96000 (in Hz). If any other value is sent an IAUDDEC1_EUNSUPPORTED error is returned.

i_coupling_channel

To indicate the element tag to select amongst multiple independent coupling channels. Used only in multi-channel libraries. Valid values are 0 or 1. If any other value is sent an IAUDDEC1_EFAIL error is returned.

Usage — Application can get a suitable set of configuration parameters from the default parameter structure and may update only a few required elements. Then an algAlloc & algInit call has to be made with these parameters.

Table 2.3 Parameters

2.2.4.2 ITTIAM_ENHAACPDEC_DynamicParams

ITTIAM_ENHAACPDEC_DynamicParams

Description — This data structure is used by application to change some dynamically configurable parameters. These parameters do not need re-initialization of the component.

Syntax

```
typedef                                    struct
ITTIAM_ENHAACPDEC_DynamicParams {
  IAUDDEC1_DynamicParams s_iauddec_dparams;
  {
    XDAS_Int32 size;
    XDAS_Int32 downSampleSbrFlag;
  }
}ITTIAM_ENHAACPDEC_DynamicParams;;
```

Parameters

size

Size of this structure in bytes.

downSampleSbrFlag

Flag to indicate downsampling for SBR. Valid values for this field are XDAS_TRUE and XDAS_FALSE. If any other value is sent an IAUDDEC1_EFAIL error is returned.

This parameter is used only HEAAC libraries.

Usage — An XDM_SETPARAMS control call has to be made with the required flag.

Table 2.4 Dynamic Parameters

2.2.4.3 ITTIAM_ENHAACPDEC_Status

ITTIAM_ENHAACPDEC_Status

Description

This data structure is used by application to get the status information of the algorithm. This structure is read only.

Syntax

```
typedef struct ITTIAM_ENHAACPDEC_Status {
  IAUDDEC1_Status  s_iauddec_status;
  {
    XDAS_Int32 size;
    XDAS_Int32 extendedError;
    XDM1_SingleBufDesc data;
    XDAS_Int32 validFlag;
    XDAS_Int32 lfeFlag;
    XDAS_Int32 bitRate;
    XDAS_Int32 sampleRate;
    XDAS_Int32 channelMode;
    XDAS_Int32 pcmFormat;
    XDAS_Int32 numSamples;
    XDAS_Int32 outputBitsPerSample;
    XDM_AlgBufInfo  bufInfo;
    XDAS_Int32 dualMonoMode;
  }
  XDAS_Int32      i_channel_mask;
  XDAS_Int32      i_channel_mode;
  XDAS_Int32      i_sbr_mode;
  XDAS_Int32      i_num_channels;
} ITTIAM_ENHAACPDEC_Status;
```

Parameters

size

Size of this structure in bytes.

extendedError

Extended error information.

Ittiam Implementation: If a fatal error occurs during process call the fatal error bit of this element is set.

data

Buffer descriptor for data passing. This is currently used only for XDM_GETVERSION command to get version information.

validFlag

Reflects the validity of this status structure. Valid values for this field are XDAS_TRUE and XDAS_FALSE.

Ittiam Implementation: Once the header decode of a stream is done this value is set to XDAS_TRUE.

lfeFlag

Flag indicating the presence of LFE channel in the output. Valid values for this field are XDAS_TRUE and XDAS_FALSE.

bitRate

Average bit rate, in bits per second.

sampleRate

Sampling frequency, in Hz. For example, if the sampling frequency is 44.1 kHz, this field will be 44100.

channelMode

Output Channel Configuration. Will be one of the IAUDIO_ChannelMode enum.

Ittiam Implementation: If the channel mode of the stream being decoded is not one of the enums, -1 will be written in this channelMode field. If an extended status structure is provided the total number of channels (including LFE channels) will be provided in the i_num_channels extended element and the channel order will be provided in the i_channel_mask extended element. Else if a base status structure is provided IAUDDEC1_EUNSUPPORTED will be returned.

pcmFormat

Output PCM Format Block/Interleaved. Only IAUDIO_INTERLEAVED is supported by the decoder.

numSamples

Number of samples in the output per channel.

outputBitsPerSample

Number of bits per output sample. Only 16 bits per output sample is supported by the decoder.

bufInfo.minNumInBufs

Minimum number of input buffers.

bufInfo.minNumOutBufs

Minimum number of output buffers.

bufInfo.minInBufSize[16]

Minimum size, in 8-bit bytes, required for each input buffer.

bufInfo.minOutBufSize[16]

Minimum size, in 8-bit bytes, required for each output buffer.

dualMonoMode

Mode to indicate type of Dual Mono. Only used in case of Dual Mono Output. Will be one of the IAUDIO_DualMonoMode enum. Only IAUDIO_DUALMONO_LR is supported by the decoder.

i_channel_mask

Channel mask required for the WAV header writing. Refer **[4]** for interpretation of values. Used for determining the channels positioning.

i_channel_mode

Channel mode of the stream being decoded. Possible cases are

0 – Mono or PS stream

1 – Stereo stream

2 – Dual-mono stream

Anything else – None of the above (possibly multi-channel)

i_sbr_mode

SBR present flag.

0 – SBR not present

1 – SBR present

Anything else – Illegal

i_num_channels

Total number of channels in the stream (including LFE channels)

Usage

This structure can be used with XDM_GETSTATUS, XDM_GETBUFINFO & XDM_GETVERSION control calls to get information about decoder status, buffers and library version.

Table 2.5 Status structure

2.2.4.4 ITTIAM_ENHAACPDEC_InArgs

ITTIAM_ENHAACPDEC_InArgs

Description

This data structure is used by application to pass information about input arguments.

Syntax

```
typedef struct ITTIAM_ENHAACPDEC_InArgs {
  IAUDDEC1_InArgs  s_iauddec_in_args;
  {
    XDAS_Int32 size;
    XDAS_Int32 numBytes;
    XDAS_Int32 desiredChannelMode;
    XDAS_Int32 lfeFlag;
  }
}ITTIAM_ENHAACPDEC_InArgs;
```

Parameters

size

Size of this structure in bytes.

numBytes

Size of input data in bytes, provided to the algorithm for decoding.

desiredChannelMode

Desired Channel Configuration.

Ittiam Implementation: Valid values are IAUDIO_1_0 to down mix the stream to mono, or IAUDIO_2_0 to down mix streams greater than 2 channels or interleave mono streams to stereo, or -1 to retain the stream's native channel configuration.

lfeFlag

Flag indicating whether LFE channel data is desired in the output.

Ittiam Implementation: Valid values are XDAS_FALSE to not provide LFE in output, or -1 to retain whatever the stream has.

Usage — This structure is sent along with the process call to provide information on number of input bytes and desired channel mode.

Table 2.6 Input Arguments

2.2.4.5 ITTIAM_ENHAACPDEC_OutArgs

ITTIAM_ENHAACPDEC_OutArgs

Description — This data structure is used by application to pass information about output arguments.

Syntax

```
typedef struct ITTIAM_ENHAACPDEC_OutArgs
{
  IAUDDEC1_OutArgs s_iauddec_outargs;
  {
    XDAS_Int32 size;
    XDAS_Int32 extendedError;
    XDAS_Int32 bytesConsumed;
    XDAS_Int32 numSamples;
    XDAS_Int32 channelMode;
    XDAS_Int32 lfeFlag;
    XDAS_Int32 dualMonoMode;
    XDAS_Int32 sampleRate;
  }
  XDAS_Int32     i_init_done;
  XDAS_Int32     i_exec_done;
  XDAS_Int32     i_ittiam_err_code ;
  XDAS_Int32     i_output_bits_per_sample;
  XDAS_Int32     i_num_channels;
  XDAS_Int32     i_channel_mask;
} ITTIAM_ENHAACPDEC_OutArgs;.
```

Parameters

size

Size of this structure in bytes.

extendedError

Extended error information.

Ittiam Implementation: If a fatal error had occurred during process call the fatal error bit of this element is set.

bytesConsumed

Number of bytes consumed during the process() call.

numSamples

Number of output samples per channel.

channelMode

Output Channel Configuration. Will be one of the IAUDIO_ChannelMode enum.

Ittiam Implementation: If the channel mode of the stream being decoded is not one of the enums, -1 will be written in this channelMode field. If an extended status structure is provided the total number of channels (including LFE channels) will be provided in the i_num_channels extended element and the channel order will be provided in the i_channel_mask extended element. Else if a base status structure is provided IAUDDEC1_EUNSUPPORTED will be returned.

lfeFlag

Flag indicating the presence of LFE channel in the output. Valid values for this field are XDAS_TRUE and XDAS_FALSE.

dualMonoMode

Mode to indicate type of Dual Mono. Only used in case of Dual Mono Output. Will be one of the IAUDIO_DualMonoMode enum. Only IAUDIO_DUALMONO_LR is supported by the decoder.

sampleRate

Sampling frequency, in Hz. For example, if the sampling frequency is 44.1 kHz, this field will be 44100.

i_init_done

Flag to indicate header decode complete, so that output status parameters can be used.

i_exec_done

Flag to indicate when execution is completed.

Here as there is no output buffering immediately after an `XDM_FLUSH` command, this flag will be set. If a default structure is sent, the `process()` call can be stopped immediately after input is over.

i_ittiam_err_code

Detailed output error code specific. For error handling refer **Section 3.3**

i_output_bits_per_sample

	Number of bits per output sample. Only 16 bits per output sample is supported by the decoder.
	i_num_channels Total number of channels in the stream (including LFE channels).
	i_channel_mask Channel mask required for the WAV header writing. Refer **[4]** for interpretation of values. Used for determining the channels positioning.
Usage	This structure is got along after a process call with useful information about the decoder's status.

Table 2.7 Output Arguments

3. API Integration

3.1 API Integration steps

The steps of integrating XDM API in a system are described. Please go through the test bench files available to see the example of these recommended steps. **Figure 3-1** gives an overview of these steps.

1. Get the IMOD function vector table or the table of all functional interfaces available for the component. This can found in the imodule_vendor.h file.
2. Make a query about the memory requirements from the algorithm, make those allocations and then inform algorithm to initialize it for further processing. At the end of this step, application will have a handle to make all further calls to the algorithm. In detail:
    a. Get the number of memory tables required (**algNumAlloc**).
    b. Allocate a memory-table for those many memory records and get the detailed requirement for all of them from the algorithm (**algAlloc**).
    c. Allocate those memory tables as per their requirement. Pointer to the first memory block becomes the handle for the component.
    d. Set configuration parameters for the algorithm as available in ITTIAM_ENHAACPDEC_Params. Use ENHAACPDECODER_ITTIAM_PARAMS to get the benefit of default values.
    e. Inform the algorithm with the record of allocated memory tables so that it can initialize itself (**algInit**).

    Allocation of memory is application specific and it can be done differently.
3. Get input /output buffer requirement from the algorithm by making call to **Control** with **XDM_GETSTATUS** or **XDM_GETBUFINFO**. If xhandle is the handle for the algorithm, then pointer to control function is xhandle->fxns->control.
4. Allocate memory for all input/output buffers.
5. Get appropriate input data and call the **Process** function for main processing of the algorithm. Once the output number of samples is non zero the stream information can be got by making call to **control** API with the **XDM_GETSTATUS** command.
6. Set configurable parameters (ITTIAM_ENHAACPDEC_DynamicParameters) and call **Control** in between process calls if needed.
7. Keep calling **Process** till data to be processed lasts.
8. At the end get all memory tables information from the algorithm (**algFree**) and free them.

3.2 Flow graph for API integration

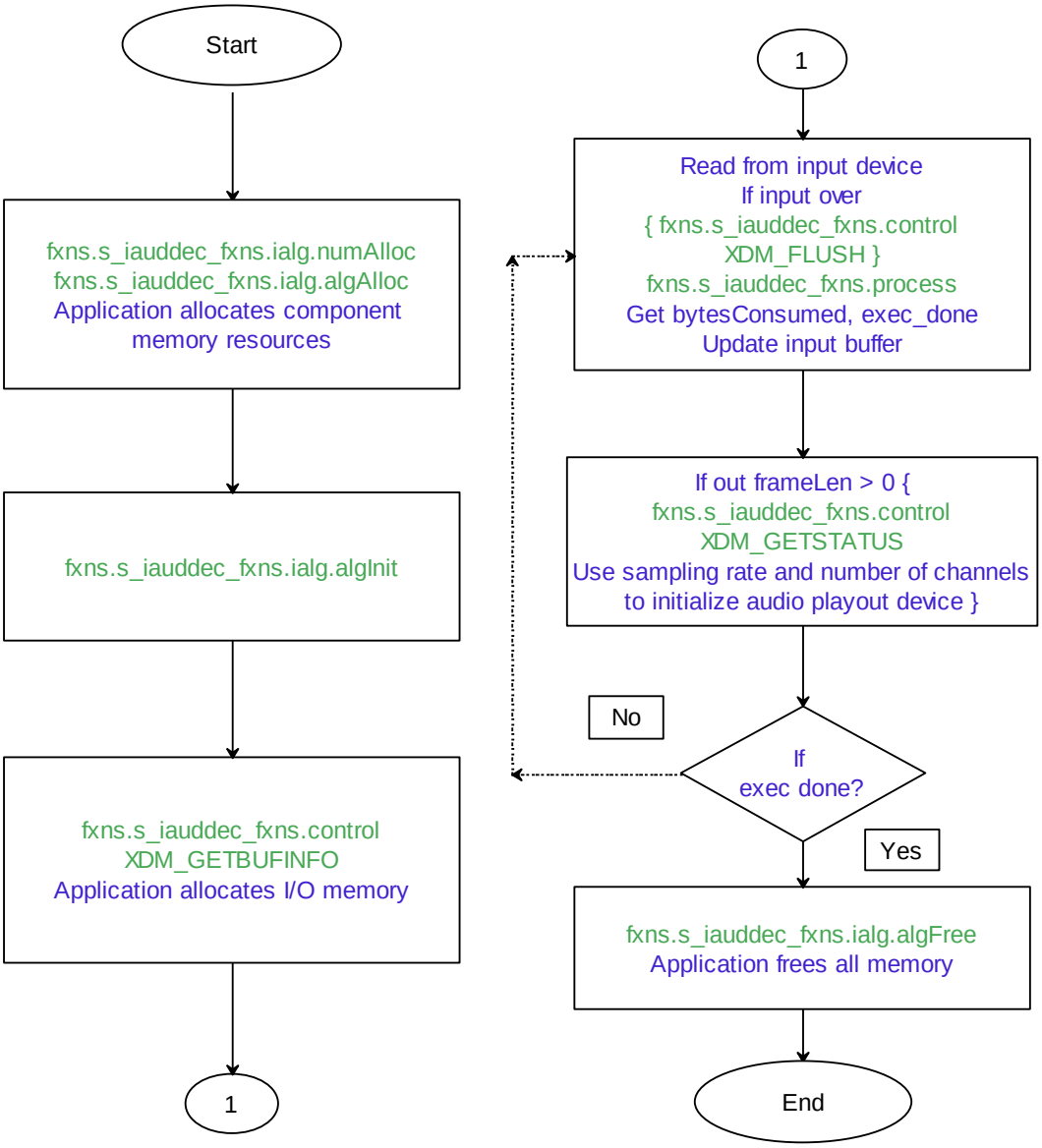

Figure 3-1 Flow-chart for Enhanced aacPlus Decoder API integration

3.3 Error Handling

The decoder algorithm signals error conditions to the sample application through error-codes. The complete listing of error codes and the handling procedure are listed down in following sections.

These error-codes are returned in ITTIAM_ENHAACPDEC_OutArgs->i_ittiam_err_code.

3.3.1 Initialization non-fatal error codes

Initialization non-fatal error codes	
0x00001000	IA_ENHAACPLUS_DEC_INIT_NONFATAL_BOTH_16AND08OUT
0x00001001	IA_ENHAACPLUS_DEC_INIT_NONFATAL_NO_UPSAMPLING
0x00001002	IA_ENHAACPLUS_DEC_INIT_NONFATAL_HEADER_NOT_AT_START

Table 3.1 Initialization non-fatal error codes

IA_ENHAACPLUS_DEC_INIT_NONFATAL_BOTH_16AND08OUT

It is returned when both out08khz flag and out16khz flag are set. In this case out08khz flag is set to 0 and out16khz flag is kept as it is.

IA_ENHAACPLUS_DEC_INIT_NONFATAL_NO_UPSAMPLING

It is returned when sampling rate specified in the header is 8 kHz and out16khz flag is set. This scenario means that the decoder should up sample the output from 8 kHz to 16 kHz, but is not supported. In this case out16khz flag is reset to 0.

IA_ENHAACPLUS_DEC_INIT_NONFATAL_HEADER_NOT_AT_START

It is returned when header information is not present in the beginning of the file.

3.3.2 Initialization fatal error codes

Initialization fatal error codes	
0xFFFF9000	IA_ENHAACPLUS_DEC_INIT_FATAL_DEC_INIT_FAIL
0xFFFF9001	IA_ENHAACPLUS_DEC_INIT_FATAL_EO_INPUT_REACHED
0xFFFF9002	IA_ENHAACPLUS_DEC_INIT_FATAL_STREAM_CHAN_GT_MAX

Table 3.2 Initialization fatal error codes

IA_ENHAACPLUS_DEC_INIT_FATAL_DEC_INIT_FAIL

It is returned when the header decode is unsuccessful.

IA_ENHAACPLUS_DEC_INIT_FATAL_EO_INPUT_REACHED

It is returned when end of input is reached in the initialization phase itself.

IA_ENHAACPLUS_DEC_INIT_FATAL_STREAM_CHAN_GT_MAX

It is returned when the number of channels in the input stream is greater than what user specified.

3.3.3 Execution non-fatal error codes

Execution non-fatal error codes	
IA_ENHAACPLUS_DEC_EXE_NONFATAL_ADTS_SYNC_LOST	0x00001800
IA_ENHAACPLUS_DEC_EXE_NONFATAL_SBR_TURNED_OFF	0x00001801
IA_ENHAACPLUS_DEC_EXE_NONFATAL_SBR_TURNED_ON	0x00001802
IA_ENHAACPLUS_DEC_EXE_NONFATAL_ADTS_HDR_CRC_FAIL	0x00001803

Table 3.3 Execution non-fatal error codes

IA_ENHAACPLUS_DEC_EXE_NONFATAL_ADTS_SYNC_LOST

It is a non-fatal error returned when ADTS sync word is not found at expected place between two frames decodes. This is only an indication of some junk data present in the bit stream.

IA_ENHAACPLUS_DEC_EXE_NONFATAL_SBR_TURNED_OFF

It is a non-fatal error returned when SBR data is not present in the current frame while it was present in the previous frame. If the decoder was initialized with the i_auto_sbr_upsample as '0' then the application has to be aware that from this frame onwards the sampling rate of the output will be half as compared to previous frame. (The number of output samples would also be half).

IA_ENHAACPLUS_DEC_EXE_NONFATAL_SBR_TURNED_ON

It is a non-fatal error returned when SBR data is present in the current frame while it was not present in the previous frame. In this case the application has to be aware that from this frame onwards the sampling rate of the output will be double as compared to previous frame. (The number of output samples would also be double).

IA_ENHAACPLUS_DEC_EXE_NONFATAL_ADTS_HDR_CRC_FAIL

This error code has been added for a future support of ADTS CRC check the current decoder would never return this error code.

3.3.4 Execution fatal error codes

The possible configuration fatal error codes generated as a part of the API as listed in. The decoder must be re-instantiated with appropriate correction in case of fatal errors.

Execution fatal error codes	
IA_ENHAACPLUS_DEC_EXE_FATAL_DECODE_FRAME_ERROR	0xFFFF9800
IA_ENHAACPLUS_DEC_EXE_FATAL_INVALID_CODE_BOOK	0xFFFF9801
IA_ENHAACPLUS_DEC_EXE_FATAL_PREDICTION_DATA_PRESENT	0xFFFF9802
IA_ENHAACPLUS_DEC_EXE_FATAL_UNIMPLEMENTED_CCE	0xFFFF9803
IA_ENHAACPLUS_DEC_EXE_FATAL_GAIN_CONTROL_DATA_PRESENT	0xFFFF9804
IA_ENHAACPLUS_DEC_EXE_FATAL_TNS_RANGE_ERROR	0xFFFF9805
IA_ENHAACPLUS_DEC_EXE_FATAL_TNS_ORDER_ERROR	0xFFFF9806
IA_ENHAACPLUS_DEC_EXE_FATAL_INSUFFICIENT_INPUT_BYTES	0xFFFF9807
IA_ENHAACPLUS_DEC_EXE_FATAL_ELE_INSTANCE_TAG_NOT_FOUND	0xFFFF9808
IA_ENHAACPLUS_DEC_EXE_FATAL_EXCEEDS_SFB_TRANSMITTED	0xFFFF9809
IA_ENHAACPLUS_DEC_EXE_FATAL_EXCEEDS_MAX_HUFFDEC_VAL	0xFFFF980A
IA_ENHAACPLUS_DEC_EXE_FATAL_CHANGED_ADTS_SF	0xFFFF980B

Table 3.4 Execution fatal error codes

IA_ENHAACPLUS_DEC_EXE_FATAL_DECODE_FRAME_ERROR

It is a fatal error returned when an error has occurred in the bit-stream decoding at other than Huffman or bit-stream decode stages. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_INVALID_CODE_BOOK

It is a fatal error returned when error has occurred in the bit-stream decoding at Huffman or bit-stream decoding stages. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_PREDICTION_DATA_PRESENT

It is a fatal error returned in case LTP object type (Prediction) is signaled in the bit-stream. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_UNIMPLEMENTED_CCE

It is a fatal error returned if Channel Coupling Element is signaled in bit-stream and is not implemented in the decoder. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_GAIN_CONTROL_DATA_PRESENT

It is a fatal error returned if gain control data is signaled in bit-stream and is not implemented in the decoder. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_TNS_RANGE_ERROR

It is a fatal error returned if TNS range specified is more than the MPEG standard specifications . The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_TNS_ORDER_ERROR

It is a fatal error returned if TNS order specified is more than the MPEG standard specifications. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_INSUFFICIENT_INPUT_BYTES

It is a fatal error returned if the decoder does have enough bits for decoding the frame. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_ELE_INSTANCE_TAG_NOT_FOUND

It is a fatal error returned if there is a mismatch of element instance tag i.e. the instance tag in the current element doesn't match any previous elements instance tag. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_EXCEEDS_SFB_TRANSMITTED

It is a fatal error returned when the number of scale factor bands exceeds the scale factor band boundary indicated while decoding Huffman codebook sectioning information. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_EXCEEDS_MAX_HUFFDEC_VAL

It is a fatal error returned if the Huffman decoded value is greater than the maximum Huffman decoded value of 8191. The decoder needs to be re-initialized with correct bit-stream.

IA_ENHAACPLUS_DEC_EXE_FATAL_CHANGED_ADTS_SF

It is a fatal error returned when the sampling frequency changes within the input ADTS stream. The decoder needs to be re-initialized with correct bit-stream.

3.3.5 Fatal error code

Execution fatal error codes	
IAUDDEC1_EFAIL	-1
IAUDDEC1_EUNSUPPORTED	-3

Table 3.5 General fatal error code

3.3.5.1 IAUDDEC1_EFAIL

It is a fatal error returned by following functions in the different scenarios listed below:

ITTIAM_ENHAACPDEC_Fxns.s_iauddec_fxns.ialg.algInit

This function returns IAUDDEC1_EFAIL in the following scenarios:

- `(IALG_Handle)handle->fxns->implementationId` is not the address of IMOD function vector table global structure
- `params->s_iauddec_params.size` is not `sizeof(ITTIAM_ENHAACPDEC_Params) or sizeof(IAUDDEC1_Params)`
- `memTab` base pointers are `NULL` or not don't match the alignment requirement
- One of the below listed parameters is not valid. Refer Section 2.2.4.1 for details on what are the valid values for each of these parameters.
  - `params->i_flag_08khz_out`
  - `params->i_flag_16khz_out`
  - `params->i_mp4_flag`
  - `params->i_coupling_channel`

ITTIAM_ENHAACPDEC_Fxns.s_iauddec_fxns.process

This function returns IAUDDEC1_EFAIL in the following scenarios:

- `(IALG_Handle)handle->fxns->implementationId` is not the address of IMOD function vector table global structure
- A previous call of `(IALG_Handle)handle->fxns->algMoved` has failed because of `memTab` base pointers were `NULL` or not did not match the alignment requirement
- Input or output buffer pointer is `NULL`
- Inargs or Outargs pointer is `NULL`
- `inargs->s_iauddec_in_args.size` is not `sizeof(ITTIAM_ENHAACPDEC_InArgs)`.
- `outargs->s_iauddec_out_args.size` is not `sizeof(ITTIAM_ENHAACPDEC_OutArgs)` or `sizeof(IAUDDEC1_OutArgs)`
- Input or output buffer number and sizes are less than the ones got in `status.s_iauddec_status.bufInfo` with `XDM_GETBUFINFO control()` call.
- More than one process call is made after calling `XDM_FLUSH control` call
- The output buffer size is less than what is required to fill in output generated.
- The decoder has faced a fatal error in decoding. In this case `outargs->i_ittiam_err_code` can be used to get the detailed error code (Refer section 3.3

ITTIAM_ENHAACPDEC_Fxns. s_iauddec_fxns.control

This function returns IAUDDEC1_EFAIL in the following scenarios:

- `(IALG_Handle)handle->fxns->implementationId` is not the address of IMOD function vector table global structure
- A non standard command is sent

- Status or dparams pointer is `NULL`
- `status->s_iauddec_status.size` is not `sizeof(ITTIAM_ENHAACPDEC_Status)` or `sizeof(IAUDDEC1_Status)`
- `dparams->s_iauddec_dynamic_params.size` is not `sizeof(ITTIAM_ENHAACPDEC_DynamicParams)`
- The `dparams->s_iauddec_dynparams.downSampleSbrFlag` is not `XDAS_FALSE or XDAS_TRUE` for the XDM_SETPARAMS command.
- `status->s_iauddec_status.data.bufSize` is < 64 or `status->s_iauddec_status.data.buf` is `NULL`

3.3.5.2 IAUDDEC1_EUNSUPPORTED

It is a fatal error returned by following functions in the different scenarios listed below:

ITTIAM_ENHAACPDEC_Fxns.s_iauddec_fxns.ialg.algInit

This function returns IAUDDEC1_EUNSUPPORTED in the following scenarios:

- One of the below listed parameters is not valid. Refer Section 2.2.4.1 for details on what are the valid values for each of these parameters.
  - `params->s_iauddec_params.outputPCMWidth`
  - `params->s_iauddec_params.pcmFormat`
  - `params->s_iauddec_params.dataEndianness`
  - `params->i_sampfreq`

ITTIAM_ENHAACPDEC_Fxns.s_iauddec_fxns.process

This function returns IAUDDEC1_EUNSUPPORTED in the following scenarios:

- The `desiredChannelMode` and the desired `lfeFlag` are not -1 and not
  - `desiredChannelMode` = `IAUDIO_1_0` & `lfeFlag` = `XDAS_FALSE` or
  - `desiredChannelMode` = `IAUDIO_2_0` & `lfeFlag` = `XDAS_FALSE` or
  - `desiredChannelMode & lfeFlag` same as the stream's native channel configuration.
- The `channelMode` of the stream is not one of the `IAUDIO_ChannelMode` enum and only a base outargs structure was provided.

ITTIAM_ENHAACPDEC_Fxns.s_iauddec_fxns.control

This function returns IAUDDEC1_EUNSUPPORTED in the following scenarios:

- The `channelMode` of the stream is not one of the `IAUDIO_ChannelMode` enum and only a base outargs structure was provided.

4. Reference

[1] SPRU360C - TMS320 DSP Algorithm Standard API Reference

[2] SPRUEC8 provided by TI. XDM user guide.

[3] xDM html documentation [part of XDAIS installation at <install_path>\docs\html\index.html].

[4] Microsoft wav extensible format. http://www.microsoft.com/whdc/device/audio/multichaud.mspx