ENHANCED AACPLUS Encoder

XDM API

Document Name	IA-HEAACv2-Enc-XDM
Version	1.0
Date	September 28, 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	September 28, 2008	Original

Contents

Figures

Tables

1. Introduction

1.1 Motivation

HE-AAC v2 (High Efficiency Advanced Audio Coding Version 2) also known as Enhanced aacPlus is a popular audio coding technique recommended by MPEG (Moving Picture Experts Group) committee. SBR (Spectral Bandwidth Replication) and PS (Parametric Stereo) are the tools used in combination with the AAC general audio codec resulting in Enhanced aacPlus. It provides significant increase in coding gain. In SBR, the high-band, i.e. the high frequency part of the spectrum is replicated using the low-band. The bit-rate is far below the bit-rate required when using conventional AAC coding. This translates into better quality at lower bit-rates.

This document describes the **A**pplication **P**rogram **I**nterface for the Enhanced aacPlus Encoder. It also addresses the knowledge requirements of developers to integrate different components of their system with Ittiam Enhanced aacPlus encoder 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”. Beyond that, this doc details the changes in interface data structures done for this component. If there is any change in the functionality of the interface functions, that will also be detailed.

Most of the information about API is given assuming XDAIS or XDM related interface is used. Both component specific information and generic information is given in this document. Thereby this document serves as a complete reference to XDM 1.0 API.

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

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

Glossary

API	Application Program Interface (Interface through which an application talks to functional blocks)
MPEG	Moving Picture Experts Group
AAC	Advanced Audio Coding
SBR	Spectral Band Replication
PS	Parametric stereo
aacPlus	AAC plus SBR
Enhanced aacPlus	AAC Plus with PS feature.

2. Interface Data Structures

In this section the list of function interfaces available for this component will be described. Both the generic fields and user defined fields will be described.

2.1 Generic notes

Here are few generic notes about all interface 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 such a case is expected to fill this element properly for all structures before communicating with the component. For the same reason, in control call, algorithm doesn't update "size" element in any case.
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 `EAACPLUSENCODER_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]** and **[3]**.

2.2 API of Enhanced aacPlus Encoder

2.2.1 Functional Interface

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

Syntax	`typedef struct ITTIAM_EAACPLUSENC_Fxns {`
	`  IAUDENC1_Fxns s_iaudenc_fxns;`
	`} ITTIAM_EAACPLUSENC_Fxns;`	

Parameters	Functions	Operations

	s_audenc_fxns.ialg	
	Implementation Id	Unique pointer that identifies the module implementing this interface.
	AlgActivate	NOT IMPLEMENTED.
	AlgAlloc	Apps call this to query the algorithm about its memory requirements.
	AlgControl	NOT IMPLEMENTED.
	AlgDeactivate	NOT IMPLEMENTED.
	AlgFree	Query algorithm for memory to free when removing an instance.
	AlgInit	Apps call this to allow the algorithm to initialize memory requested via algAlloc().
	AlgMoved	Apps call this whenever an algorithm's object or any pointer parameters are moved in real-time.
	AlgNumAlloc	Query algorithm for number of memory requests.
	s_iaudenc_fxns.	
	process	Main process call for Enhanced aacPlus Encoder operation. It takes single buffers for input and output. In case of any fatal errors it returns XDM_EFAIL or XDM_EUNSUPPORTED. Codec specific error code got from ITTIAM_EAACPLUSENC_OutArgs-> i_ittiam_err_code. For details refer **Section 3.3** on error handling.
	control	
	Commands	Operations
	XDM_GETSTATUS	Query algorithm to fill Status structure. For description of elements and their valid values refer section **2.2.4.3**
	XDM_SETPARAMS	Set run time dynamic parameters. For description of elements and 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 encoder returns to the state it was just after algInit().
	XDM_SETDEFAULT	Restore the algorithm's internal state to its original, default values. ITTIAM IMPLEMENTATION: Restore the dynamic parameters stored in the internal state to their original default values.

	XDM_FLUSH	Execute done flag is set inside the encoder as a result of which encoder completes its operation.
	XDM_GETBUFINFO	Query 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 signed 16 bit little endian PCM samples only.

2.2.2.2 Output Format

The generated output is compressed audio data in byte format.

2.2.3 Default Parameters

EAACPLUSENCODER_ITTIAM_PARAMS	
Description	This contains the default initialization parameters for the component.
Syntax	typedef struct ITTIAM_EAACPLUSENC_Params { IAUDENC1_Params s_iaudenc_params; /* Extended params */ XDAS_Int32 noChannels; XDAS_Int32 aacClassic; XDAS_Int32 psEnable; XDAS_Int32 dualMono; XDAS_Int32 downmix; XDAS_Int32 useSpeechConfig; XDAS_Int32 fNoStereoPreprocessing; XDAS_Int32 invQuant; XDAS_Int32 useTns; XDAS_Int32 use_ADTS; XDAS_Int32 use_ADIF; XDAS_Int32 full_bandwidth; XDAS_Int32 i_channels_mask;

	XDAS_Int32 i_num_coupling_chan; XDAS_Int32 write_program_config_element; } ITTIAM_EAACPLUSENC_Params;	
Parameters	Names	Values
	s_iaudenc_params.	
	size	sizeof(ITTIAM_EAACPLUSENC_Params)
	sampleRate	48000
	bitRate	128000
	channelMode	IAUDIO_2_0
	dataEndianness	XDM_LE_16
	encMode	IAUDIO_CBR
	inputFormat	IAUDIO_INTERLEAVED
	inputBitsPerSample	16
	maxBitRate	144000(576000 for AAC only build)
	dualMonoMode	IAUDIO_DUALMONO_LR
	crcFlag	XDAS_FALSE
	ancFlag	XDAS_FALSE
	lfeFlag	XDAS_FALSE
	Extended Params	
	noChannels	2
	aacClassic	0(1 for AAC only build)
	psEnable	0(1 for HEAACv2 build)
	dualMono	0
	downmix	0
	useSpeechConfig	0
	fNoStereoPreprocessing	0
	invQuant	2(0 for AAC only build)
	useTns	1
	use_ADTS	1
	use_ADIF	0
	full_bandwidth	0
	Below given params are applicable only to multichannel build	
	i_channels_mask	0x0(0x3F for multichannel build)
	i_num_coupling_chan	0
	write_program_config_element	0
Usage	Application gets a set of suitable set of configuration parameters and may need to update only few of them.	

Table 2.2 Default Parameters

2.2.4 Data structures

2.2.4.1 ITTIAM_EAACPLUSENC_Params

<table>
<tr><th colspan="3">ITTIAM_EAACPLUSENC_Params</th></tr>
<tr><td>Description</td><td colspan="2">This data structure is used by application to convey the chosen configuration to the algorithm so that later it can take appropriate actions.</td></tr>
<tr><td>Syntax</td><td colspan="2"><pre>typedef struct ITTIAM_EAACPLUSENC_Params
{
    IAUDENC1_Params s_iaudenc_params;
      /* Extended params */
      XDAS_Int32 noChannels;
      XDAS_Int32 aacClassic;
      XDAS_Int32 psEnable;
      XDAS_Int32 dualMono;
      XDAS_Int32 downmix;
      XDAS_Int32 useSpeechConfig;
      XDAS_Int32 fNoStereoPreprocessing;
      XDAS_Int32 invQuant;
      XDAS_Int32 useTns;
      XDAS_Int32 use_ADTS;
      XDAS_Int32 use_ADIF;
      XDAS_Int32 full_bandwidth;
      XDAS_Int32 i_channels_mask;
      XDAS_Int32 i_num_coupling_chan;
      XDAS_Int32 write_program_config_element;
} ITTIAM_EAACPLUSENC_Params;</pre></td></tr>
<tr><td rowspan="8">Parameters</td><td>Names</td><td>Values</td></tr>
<tr><td colspan="2">s_iaudenc_params.</td></tr>
<tr><td>size</td><td>Size of this structure in bytes. Should be either sizeof(IAUDENC1_Params), in case of base structure, or sizeof(ITTIAM_ EAACPLUSENC _Params), in case of extended structure.</td></tr>
<tr><td>sampleRate</td><td>Sampling Frequency in Hz.</td></tr>
<tr><td>bitRate</td><td>Average bit rate, in bits per second.</td></tr>
<tr><td>channelMode</td><td>Input Channel Mode. Only IAUDIO_1_0 and IAUDIO_2_0 are supported.</td></tr>
<tr><td>dataEndianness</td><td>Endianness of input data. Only XDM_LE_16 is supported.</td></tr>
<tr><td>encMode</td><td>Encoding mode. Only</td></tr>
</table>

		IAUDIO_CBR is supported.
	inputFormat	Input format block/interleaved. Only IAUDIO_INTERLEAVED is supported.
	inputBitsPerSample	Number of bits per input sample. Only 16 bits per sample is supported.
	maxBitRate	Maximum bit rate supported.
	dualMonoMode	Not supported in stereo build
	crcFlag	Flag indicating whether the encoder should insert CRC bits into the bitstream or not. Valid values are XDAS_TRUE and XDAS_FALSE.
	ancFlag	Ancillary Data Flag. Should compulsorily be set to XDAS_FALSE.
	lfeFlag	Flag indicating whether LFE channel data is present or not in the input. Should compulsorily be set to XDAS_FALSE.
	Extended Params	
	noChannels	Total number of channels to be processed. Valid values are 1 or 2.
	aacClassic	Usage of AAC Only mode. This has to be set to 1 for AAC only encoding mode. Not applicable for AAC-LC build
	psEnable	Flag to enable ps encoding.Not applicable to AAC LC and HEAAC builds. Valid values are 0 or 1.
	dualMono	Flag to enable Dual mono encoding. Applicable only to multichannel build.
	downmix	Option to enable downmix. Valid values are 0 or 1.
	useSpeechConfig	Use speech configuration flag. If this is set to 1 speech configuration is enabled, if 0 it is disabled. Not applicable for AAC-LC build.
	fNoStereoPreprocessing	Stereo Preprocessing flag. 1 to disable Stereo Preprocessing. Not applicable for mono files. Only applicable when sampleRate < 24000 Hz and bitRate < 60000 bps.
	invQuant	Inverse Quantization Level. Can be 0,1,2. 2 gives higest quality. 0 for

		lowest complexity. Not applicable for AAC-LC library.
	useTns	Flag for TNS enable. 1 for enabling TNS. 0 for disabling TNS.
	use_ADTS	Flag to enable ADTS header inclusion. Valid values are 0 and 1.
	use_ADIF	Flag to enable ADIF header inclusion. Valid values are 0 and 1.
	full_bandwidth	Flag for enabling full bandwidth If set to 1 bandwidth is set to half of sampling frequency (Full Bandwidth). 0 for adjusting bandwidth according to bit rate.
	i_channels_mask	Channel mask value which gives the bitstream elements present in the input data.Not valid for stereo build.
	i_num_coupling_chan	Number of coupling channels present in the input files. Not valid for stereo build.
	write_program_config_element	Flag to enable PCE writing. Valid values are 0 and 1.

Table 2.3 Config Parameters

2.2.4.2 ITTIAM_EAACPLUSENC_DynamicParams

ITTIAM_EAACPLUSENC_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_EAACPLUSENC_DynamicParams { IAUDENC1_DynamicParams s_iaudenc_dynamic_params; } ITTIAM_EAACPLUSENC_DynamicParams;`	
Parameters	Element	Value
	`s_iaudenc_dynamic_params.`	
	`size`	Size of this structure in bytes.
	`bitRate`	Average bit rate, in bits per second. Should be same as the value set at init time through `Params` structure.
	`sampleRate`	Sampling Frequency in Hz. Should be same as the value set at init time through `Params` structure.
	`channelMode`	Input Channel Mode. Should be same as the value set at init time through `Params` structure.
	`lfeFlag`	Flag indicating whether LFE channel data is present or not in the input.

		Should compulsorily be set to XDAS_FALSE.
	dualMonoMode	Dual mono support.
	inputBitsPerSample	Number of bits per input sample. Should be same as the value set at init time through Params structure.
Usage	Run time change in configuration parameters is not supported by this implementation of Enhaacplus Encoder.	

Table 2.4 Dynamic Parameters

2.2.4.3 ITTIAM_EAACPLUSENC_Status

ITTIAM_EAACPLUSENC_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_EAACPLUSENC_Status { IAUDENC1_Status s_iaudenc_status; } ITTIAM_EAACPLUSENC_Status;	
Parameters	Element	Value
	s_iaudenc_status.	
	size	Size of this structure in bytes.
	extendedError	Extended error information. In case of internal error, the XDM_FATALERROR or the XDM_UNSUPPORTEDPARAM bit of this element is set.
	data	Buffer descriptor for data passing. This is currently used only in XDM_GETVERSION and XDM_GETSTATUS commands to get version information. The buffer should at least be 64 bytes long. In case version information is written, the XDM_ACCESSMODE_WRITE bit of data.accessMask is set.
	validFlag	Flag indicating the validity of the Status structure. Valid values for this field are XDAS_TRUE and XDAS_FALSE.
	lfeFlag	Flag indicating whether LFE channel data is present or not in the input. Will always be set to XDAS_FALSE in this implementation.
	bitRate	Average bit rate, in bits per second.
	sampleRate	Sampling frequency, in Hz.
	channelMode	Input Channel Mode.
	encMode	Encoding mode. Will always be set to IAUDIO_CBR in this implementation.

	s_iaudenc_status.bufInfo.	
	minNumInBufs	Minimum number of input buffers.
	minNumOutBufs	Minimum number of output buffers.
	minInBufSize[16]	Minimum size, in 8-bit bytes, required for each input buffer.
	minOutBufSize[16]	Minimum size, in 8-bit bytes, required for each output buffer.
Usage	This structure can be used with XDM_GETSTATUS, XDM_GETBUFINFO and XDM_GETVERSION control() calls to get information about encoder status, buffers and library version.	

Table 2.5 Status Structure

2.2.4.4 ITTIAM_EAACPLUSENC_InArgs

ITTIAM_EAACPLUSENC_InArgs		
Description	This data structure is used by application to pass information about input arguments.	
Syntax	typedef struct ITTIAM_EAACPLUSENC_InArgs { IAUDENC1_InArgs s_iaudenc_in_args; } ITTIAM_EAACPLUSENC_InArgs;	
Parameters	Element	Value
	s_iaudenc_in_args.	
	size	Size of this structure in bytes.
	numInSamples	Number of Input Samples per Channel.
	ancData	NOT RELEVANT HERE.
Usage	This structure is sent along with the process() call to provide information on the number of valid input samples per channel in the input buffer.	

Table 2.6 Input Arguments

2.2.4.5 ITTIAM_EAACPLUSENC_OutArgs

ITTIAM_EAACPLUSENC_OutArgs	
Description	This data structure is used by application to get output arguments of the algorithm.
Syntax	typedef struct ITTIAM_EAACPLUSENC_OutArgs { IAUDENC1_OutArgs s_iaudenc_out_args; /* ITTIAM extensions */ XDAS_Int32 i_exec_done; XDAS_Int32 i_ittiam_err_code; } ITTIAM_EAACPLUSENC_OutArgs;

Parameters	Element	Value
	s_iaudenc_out_args.	
	size	Size of this structure in bytes. Should be either sizeof(IAUDENC1_OutArgs), in case of base structure, or sizeof(ITTIAM_EAACPLUSENC_OutArgs), in case of extended structure.
	extendedError	Extended error information. In case of internal error, the XDM_FATALERROR bit of this element is set.
	bytesGenerated	Number of encoded bytes generated during the process() call.
	numZeroesPadded	Number of zero samples per channel padded to the input.
	numInSamples	Number of input samples per channel consumed by the encoder.
	Extended parameters	
	i_exec_done	Flag to indicate end of execution.
	i_ittiam_err_code	Codec specific error code. For details refer Section **3.3** on Error Handling.
Usage	This structure is sent along with the process() call to get the run time output arguments of the encoder.	

Table 2.7 Output Arguments

3. API Integration

3.1 API Integration steps

Here the steps involved in integrating XDMI in a system is described. Please go through the test bench files available to see the example of these recommended steps. The process for Enhanced aacPlus Encoder is explained here. **Figure 3-1** gives an overview of these steps.

1. Get vtable or the table of all functional interfaces available for the component. For this component it is ITTIAM_EAACPLUSENC_FXNS.
2. Make a query about the memory requirements from the algorithm, make those allocations and then inform algorithm to initialize itself 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 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 algorithm (**algAlloc**).
    c. Allocate those memory tables as per their requirement. Pointer to first memory block becomes handle for the component.
    d. Set configuration parameters for algorithm as available in ITTIAM_EAACPLUSENC_Params. Use ITTIAM_EAACPLUSENC_PARAMS to get benefit of default values.
    e. Inform algorithm with record for allocated memory tables so that it can initialize itself (**algInit**).

    All of these steps are combined in **ALG_create** function provided to ease integrating and validating this component. Allocation of memory is very application specific it can be done differently.

3. Get input /output buffer requirement from 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-> s_iaudenc_fxns.control.
4. Allocate memory for all input/output buffers. One test function **allocate_audio_io_buffers** does this job.
5. Get appropriate input data and the call **Process** function for main processing of the algorithm. If xhandle is the handle for the algorithm, then pointer to **process** function is xhandle->fxns-> s_iaudenc_fxns.process.
6. Set configurable parameters (ITTIAM_EAACPLUSENC_ 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 algorithm (**algFree**) and free them. **ALG_Delete & free_audio_buffers** are example functions to complete this task.

3.2 Flow graph for API integration

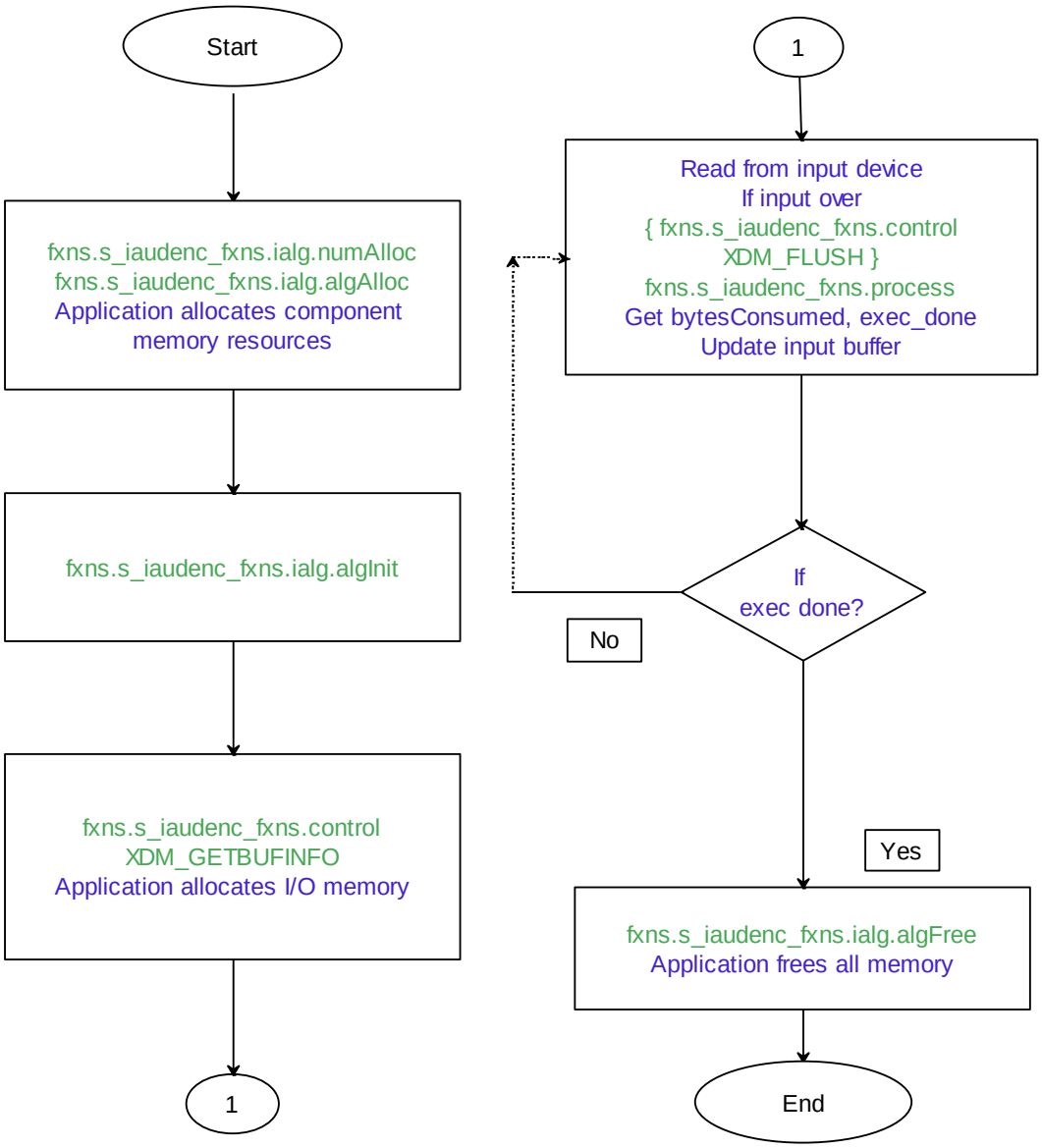

Figure 3-1 Flow-chart for Enhanced aacPlus Encoder API integration

3.3 Error Handling

The encoder 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 errors can be returned during `process()`, `ialg.algInit()`, `control()` function calls:

3.3.1 Fatal error code

Execution fatal error codes	
IALG_EFAIL	-1

Table 3.1 Fatal Error Code

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

ITTIAM_EAACPLUSENC_Fxns.s_iaudenc_fxns.ialg.algInit

This function can return fatal error in the following scenarios:

- `(IALG_Handle)handle->fxns->implementationId` is not the address of IMOD function vector table global structure.
- `memTab` base pointers are `NULL` or not don't match the alignment requirement
- `params->s_iaudenc_params.size` is not `sizeof(ITTIAM_EAACPLUSENC_Params) and also is not equal to sizeof(IAUDENC1_Params)`
- `params->s_iaudenc_params.channelMode` is not IAUDIO_1_0 or IAUDIO_2_0
- Encoder init returns a failure.

ITTIAM_EAACPLUSENC_Fxns.s_iaudenc_fxns.process

This function can return fatal error in the following scenarios:

- `(IALG_Handle)handle->fxns->implementationId` is not the address of IMOD function vector table global structure.
- A previous call of `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->s_iaudenc_in_args.size` is not `sizeof(ITTIAM_EAACPLUSENC_InArgs)` or `sizeof(IAUDENC1_InArgs)`
- `outargs->s_iaudenc_out_args.size` is not `sizeof(ITTIAM_EAACPLUSENC_OutArgs)` or `sizeof(IAUDENC1_OutArgs)`
- `sampleRate` is not 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000.
- `dataEndianness` is not `XDM_LE_16`
- `bitRate` is not in the permissible range.
- `ps_enable and aac_only flags` are both set to 1.
- `useSpeechConfig` is not 0 or 1.
- `fNoStereoPreprocessing` is not 0 or 1.
- `invQuant` is not 0, 1 or 2.
- `useTns` is not 0 or 1.

- encMode is not IAUDIO_CBR
- inputFormat is not IAUDIO_INTERLEAVED
- inputBitsPerSample is not 16

ITTIAM_EAACPLUSENC_Fxns.s_iaudenc_fxns.control

This function can return fatal error 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->s_iaudenc_status.size is not sizeof(ITTIAM_EAACPLUSENC_Status) and not equal to sizeof(IAUDENC1_Status) or status is NULL for commands XDM_GETSTATUS, XDM_GETVERSION & XDM_GETBUFINFO
- params->s_iaudenc_dynamic_params.size is not sizeof(ITTIAM_EAACPLUSENC_DynamicParams) and is not equal to sizeof(IAUDENC1_DynamicParams) or dynamic params is NULL for commands XDM_SETPARAMS & XDM_SETDEFAULT

Execution fatal error codes	
XDM_EUNSUPPORTED	-3

ITTIAM_EAACPLUSENC_Fxns.s_iaudenc_fxns.ialg.algInit

- Invalid channel mode is given in the input params.

ITTIAM_EAACPLUSENC_Fxns.s_iaudenc_fxns.control

- params->s_iaudenc_dynamic_params.inputBitsPerSample is not IAUDIO_INTERLEAVED for command XDM_SETPARAMS
- params->s_iaudenc_dynamic_params.sampleRate is not same as the sampling rate set during algInit() for command XDM_SETPARAMS
- params->s_iaudenc_dynamic_params.channelMode is not same as the channel mode set during algInit() for command XDM_SETPARAMS
- params->s_iaudenc_dynamic_params.lfeFlag is not same as the lfeFlag set during algInit() for command XDM_SETPARAMS

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].