MPEG Audio Decoder (AdecMpeg) API

Overview

Introduction

The MPEG audio decoder is a TSSA compliant module that accepts a stream of MPEG 1 layer 1 and layer 2 encoded audio at its input stream and generates a linear PCM format output stream. It is also able to handle the respective MPEG-2 bit streams. However, it decodes only the stereo channels of MPEG-2 streams. For information about the general interface philosophy, you are directed to the TSSA software architecture documentation.

The public programmers interface of the decoder is the file tmolAdecMpeg.h. This TriMedia library does not support a non-streaming interface. Therefore, no AL header file is made public.

Use of either of these decoders may require a patent license, as the MPEG audio coding standards are covered by patents held by various companies.

MPEG Compliancy

The decoder is capable of decoding all Layer 1and Layer 2 bit streams except for bit streams using the free data rate format. Such bit streams cause an error message. The decoder is also not performing de-emphasis. It, however, indicates if emphasis is applied to MPEG bit stream via the progress callback function when the appropriate flag is installed.

Inputs and Outputs

The decoder has one input and two outputs. The input is an MPEG 1 encoded bit stream. The first output is stereo 16 bit linear PCM audio data, as described by a TSA packet. Stereo 16 bit is the only supported output format. The sample rate can be 32k, 44.1k, or 48k, as described by the MPEG specification. The second will support IEC601937 formatted data, or a headphone mix, in the future.

Real Time Behavior

This section describes some issues of using the decoder in a real time application as buffering, time stamping, and synchronization.

Input/Output Buffering

The MPEG-1 audio decoder accepts TSSA data packets of the type atfMpeg and sends out packets of the type atfLinearPCM and the subtype apfStereo16. On its input side the decoder implements a flexible buffer management. It accepts packets of any size. On the output side, however, it accepts only packets that can accommodate at least one frame of decoded audio which is 284 samples for Layer 1 and 1152 samples for Layer 2. The decoder sends the output packet when it is filled with one decoded audio frame. It does not try to fill the rest of the packet with data from successive frames.

Time Stamps

The MPEG audio decoder is capable of attaching time stamps to the PCM data packets which are copied from the incoming MPEG packets. It is ensured that the time stamps are assigned to the correct PCM packets.

Synchronization

After the start function of the decoder has been called the decoder can either be in sync or out of sync. It reports a change of this state through the progress function if the progress flag ADEC_MPEG1_PROG_REPORT_FIND_SYNC is installed. Whenever the decoder is not in sync it is not producing audio output. It loses the synchronization, when settings in the MPEG headers change, the header is invalid, the distance to the next frame is incorrect, or the optional CRC is incorrect. In the latter three cases an error is reported via the error callback function. In all cases the progress function is called if the above mentioned progress flag is installed.

The decoder does not perform any muting or block repeating when it loses sync. It is up to downstream components to implement features like that.

Errors

The errors reported by the MPEG decoder are all defined in tmolAdecMpeg.h. The base value of these errors is 0x140A0000, as defined in tmLibappErr.h.

The user can install a TSA standard error callback function, and the decoder will call this if it encounters errors while decoding the bit stream. In that case, the errorCode will be one of the values defined in tmolAdecMpeg.h. Errors reported by the error function are not fatal, and processing will continue as the decoder attempts to recover from the error.

Apart from the standard TSSA errors that are defined in tmLibappErr.h the following component specific errors can occur during the execution of the start function:

`ADEC_MPEG1_ERR_INVALID_HEADER`	The ID bit in the MPEG header equals zero.
`ADEC_MPEG1_ERR_FREE_FORMAT_NOT_SUPPORTED`	MPEG bit stream does not have a specified data rate. This mode is not supported.
`ADEC_MPEG1_ERR_LAYER3_NOT_SUPPORTED`	Decoder can only handle Layer 1 and 2 bit streams.
`ADEC_MPEG1_ERR_CRC_FAILED`	The calculation of the cyclic redundancy check failed. This is an indication for a corrupted bit stream and/or transmission errors.
`ADEC_MPEG1_ILLEGAL_FRAME_LENGTH`	The decoder read more bits than permitted by the standard to decode the last frame. This is an indication that either the encoder did not work properly or that transmission errors occurred.

Progress

The user can install a TSA standard progress callback function. The decoder will use this in several cases.

1. To report a change in format, per standard TSSA behavior. The defaults handle this.
2. To report a change in format to the user. In this case, the progress flag is `ADEC_MPEG1_PROG_REPORT_FORMAT`, and the progress argument description field is a pointer to a data structure of the type `tmAdecMpegFormat_t`.
3. To report the state of the decoder while decoding. In this case, the progress flag is `ADEC_MPEG1_PROG_REPORT_FIND_SYNC`. The decoder reports its state in the description field of the progress arguments struct. It contains a pointer to an integer. The integer value is either `DECODER_NOT_IN_SYNC` or `DECODER_IN_SYNC`. Note that the progress function only reports transitions between these two states.
4. To report that a frame is decoded successfully. In this case, the progress flag is `ADEC_MPEG1_PROG_REPORT_EVERY_FRAME`. This can be used to count frames or to do some performance measurements.

Configuration

Although the decoder does export the standard configuration function, no configuration changes are supported.

Using the MPEG Audio Decoder API

The TriMedia MPEG Audio decoder API is contained within the archived application library libtmAdecMpeg.a. For OL layer applications, you must include the tmolAdecMpeg.h header file. AL layer operation is not supported.

The OL Layer

The operating system layer only supports data streaming operation. A diagram of the typical flow of control is shown in Figure 1-1.

The capabilities of the component should be obtained using `tmolAdecMpegGetCapabilities`. This information will be used by the format manager to ensure that the two instances being connected together are compatible. An instance of the audio decoder should be obtained using `tmolAdecMpegOpen`. InOutDescriptors which connect the audio decoder to other components should be created by initializing `ptsaInOutDescriptorSetup_t` structures and calling `tsaDefaultInOutDescriptorCreate` for each connection. This function can also be used to automatically create packets which will be used to transfer data between component instances.

The pointer to the audio decoder instance setup should be obtained using `tmolAdecMpegGetInstanceSetup`. This structure should be initialized with any application specific values. The application should then call `tmolAdecMpegInstanceSetup` to configure the instance.

Data streaming can then be initiated by calling `tmolAdecMpegStart`. Coded audio packets to be decoded are obtained using the datain call back function which is provided in the tsaDefaults library. An output packet will be obtained using the dataout call back function and this will be used to store the decoded audio data.

The application can terminate data streaming using `tmolAdecMpegStop`, and release the instance using `tmolAdecMpegClose`. After the instance has been closed, the application should destroy the InOutDescriptor using the `tsaDefaultInOutDescriptorDestroy` function. This will automatically free the packets contained in the queues.

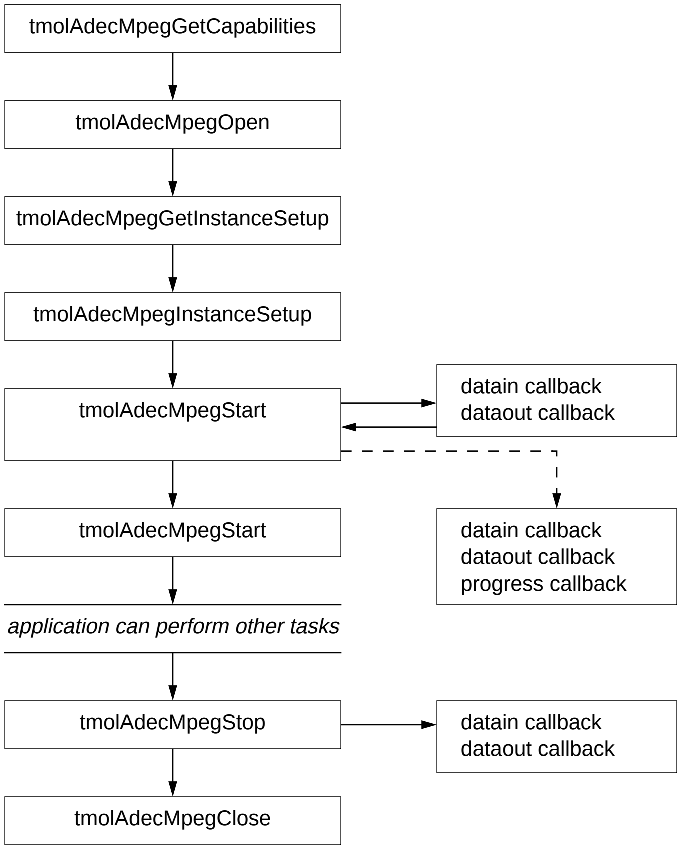

Figure 1-1 OL Layer data Streaming Flow Control

Callback Function Requirements

The following table indicates the mandatory and optional callback functions used by the MPEG audio decoder.

Table 1-1 Callback Function Requirements

Callback Function	Use
datainFunc (mandatory)	Used for data streaming to obtain full packets containing coded audio data. The tsaDefaults library provides a default function automatically.
dataoutFunc (mandatory)	Used for data streaming to obtain empty packets where decoded audio data will be stored. The tsaDefaults library provides a default function automatically.
controlFunc (mandatory)	Used to pass configuration command to the decoder. The tsaDefaults library provides a default function automatically.
progressFunc (mandatory)	Used by the decoder to report the decoders progress to the application. The tsaDefaults library provides a default function automatically.

API Data Structure Descriptions

This section describes the TriMedia MPEG-1 Layer II and Layer III audio decoder data structures.

Name	**Page**
tmolAdecMpegCapabilities_t	1-10
tmAdecMpegProgressFlags_t	1-10
tmAdecMpegMode_t	1-10
tmAdecMpegLayer_t	1-11
tmAdecMpegCopyright_t	1-11
tmAdecMpegProtection_t	1-12
tmAdecMpegPrivate_t	1-12
tmAdecMpegOriginal_t	1-11
tmAdecMpegEmphasis_t	1-12
tmAdecMpegSecOutputMode_t	1-13
tmolAdecMpegInstanceSetup_t	1-14
tmAdecMpegFormat_t	1-15

tmolAdecMpegCapabilities_t

```
typedef struct tmolAdecMpegCapabilities_t {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmolAdecMpegCapabilities_t, *ptmolAdecMpegCapabilities_t;
```

Description

Standard TSSA capabilities structure. Used by applications to find out about the inputs and outputs of the component.

tmAdecMpegProgressFlags_t

```
typedef enum {
   ADEC_MPEG1_PROG_REPORT_FORMAT       = 0x01,
   ADEC_MPEG1_PROG_REPORT_FIND_SYNC    = 0x02,
   ADEC_MPEG1_PROG_REPORT_EVERY_FRAME  = 0x04
} tmAdecMpegProgressFlags_t;
```

Description

Controls the operation of the progress callback function. An application programmer can request notification in any of these cases. These flags are used to configure the progress function behavior during instance setup. In addition to that they are also used during the data streaming. Whenever the library calls the progress function, it indicates via the in progressCode field of the progress arguments which progress flag caused the function call.

tmAdecMpegMode_t

```
typedef enum {
   ADEC_MPEG1_STEREO           = 0x00000001,
   ADEC_MPEG1_JOINT_STEREO     = 0x00000002,
   ADEC_MPEG1_DUAL_CHANNEL     = 0x00000004,
   ADEC_MPEG1_SINGLE_CHANNEL  = 0x00000008
} tmAdecMpegMode_t;
```

Description

Describes the mode of the encoded audio. This type is used in the struct `tmAdecMpegFormat_t`.

tmAdecMpegLayer_t

```
typedef enum {
   ADEC_MPEG1_LAYER1  = 0x01,
   ADEC_MPEG1_LAYER2  = 0x02,
   ADEC_MPEG1_LAYER3  = 0x03
} tmAdecMpegLayer_t;
```

Description

Describes the encoding mode of the current stream. Reported in the tmAdecMpegFormat_t structure, as found in the bit stream.

tmAdecMpegCopyright_t

```
typedef enum {
   ADEC_MPEG1_COPYRIGHT_ON   = 0x01,
   ADEC_MPEG1_COPYRIGHT_OFF  = 0x02
} tmAdecMpegCopyright_t;
```

Description

Describes the copyright state of the current stream. Reported in the tmAdecMpegFormat_t structure, as found in the bit stream.

tmAdecMpegOriginal_t

```
typedef enum {
   ADEC_MPEG1_ORIGINAL  = 0x01,
   ADEC_MPEG1_COPY      = 0x02
} tmAdecMpegOriginal_t;
```

Description

Describes the state of the “original” bit in the current stream. Reported in the `tmAdecMpegFormat_t` structure, as found in the bit stream.

tmAdecMpegProtection_t

```
typedef enum {
   ADEC_MPEG1_CRC_ON   = 0x01,
   ADEC_MPEG1_CRC_OFF  = 0x00
} tmAdecMpegProtection_t;
```

Description

Tells whether or not CRC checksum are used to protect the transmitted bit stream. Reported in the `tmAdecMpegFormat_t` structure, as found in the bit stream.

tmAdecMpegPrivate_t

```
typedef enum {
   ADEC_MPEG1_PRIVATE_ON   = 0x01,
   ADEC_MPEG1_PRIVATE_OFF  = 0x02
} tmAdecMpegPrivate_t;
```

Description

Describes the state of the “private” bit in the current stream. Reported in the `tmAdecMpegFormat_t` structure, as found in the bit stream.

tmAdecMpegEmphasis_t

```
typedef enum {
   ADEC_MPEG1_NO_EMPHASIS     = 0x01,
   ADEC_MPEG1_50_15_EMPHASIS  = 0x02,
   ADEC_MPEG1_CCITT_EMPHASIS  = 0x03,
} tmAdecMpegEmphasis_t;
```

Description

Tells a user whether or not emphasis has been applied to the current stream. Reported in the tmAdecMpegFormat_t structure, as found in the bit stream.

tmAdecMpegSecOutputMode_t

```
typedef enum {
   ADEC_MPEG1_SEC_OUT_DISABLED  = 0x01,
   ADEC_MPEG1_SEC_OUT_1937      = 0x02,
} tmAdecMpegSecOutputMode_t;
```

Description

Sets the mode of operation for the second audio output. Always `ADEC_MPEG1_SEC_OUT_DISABLED` in this release.

tmolAdecMpegInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t   defaultSetup;
    tmAdecMpegSecOutputMode_t   secondOutputMode;
} tmolAdecMpegInstanceSetup_t, *ptmolAdecMpegInstanceSetup_t;
```

Fields

defaultSetup	Pointer to the default instance setup struct, refer to tsa.h.
secondOutputMode	To allow for 1937 output. Must be `ADEC_MPEG1_SEC_OUT_DISABLED` in this release.

Description

Configure the component for operation. Standard TSSA callback functions can be provided.

tmAdecMpegFormat_t

```
typedef struct AdecMpegFormat_t {
   tmAdecMpegLayer_t          layer;
   tmAdecMpegMode_t           eMode;
   UInt32                     bitRate;
   tmAdecMpegCopyright_t      copyright;
   tmAdecMpegProtection_t     protection;
   tmAdecMpegPrivate_t        private;
   tmAdecMpegOriginal_t       original;
   tmAdecMpegEmphasis_t       emphasis;
   Float                      sampleRate;
} tmAdecMpegFormat_t;
```

Fields

layer	Encoding method, layer 1, 2, or 3.
emode	Stereo mode.
bitRate	Encoded bit rate.
copyright	Recovered from bit stream.
protection	Is CRC used? Recovered from bit stream.
private	Recovered from bit stream.
original	Recovered from bit stream.
emphasis	Recovered from bit stream.
sampleRate	Recovered from bit stream.

Description

A structure of this type is passed to progress function when the sync word is found in a bit stream. An application can use this to determine the nature of the stream.

API Function Descriptions

This section describes the TriMedia MPEG-1 Layer II audio decoder functions.

Name	**Page**
tmolAdecMpegGetCapabilities	1-17
tmolAdecMpegOpen	1-18
tmolAdecMpegClose	1-19
tmolAdecMpegGetInstanceSetup	1-20
tmolAdecMpegInstanceSetup	1-21
tmolAdecMpegStart	1-24
tmolAdecMpegStop	1-25
tmolAdecMpegInstanceConfig	1-23

tmolAdecMpegGetCapabilities

```
extern tmLibappErr_t tmolAdecMpegGetCapabilities (
   ptmolAdecMpegCapabilities_t   *pCap
);
```

Parameters

pCap	Pointer to a capabilities structure pointer.

Return Codes

`TMLIBAPP_OK`	Returned on successful completion.

Description

This function can be used to determine the capabilities of the audio decoder.

tmolAdecMpegOpen

```
extern tmLibappErr_t tmolAdecMpegOpen (
   Int   *instance
);
```

Parameters

instance	Pointer to an integer instance variable which will be used to identify the decoder in subsequent transactions.

Return Codes

`TMLIBAPP_OK`	Returned on successful completion.
`TMLIBAPP_ERR_MEMALLOC_FAILED`	Memory could not be allocated for the internal variables.

Description

Instantiates a MPEG audio decoder, and sets the instance variable to point to the audio decoder instance. Allocates memory for the instance variable

tmolAdecMpegClose

```
extern tmLibappErr_t tmolAdecMpegClose (
   Int   instance
);
```

Parameters

instance	Instance value, as returned by tmxlAdecMpegOpen

Return Codes

TMLIBAPP_OK	Returned on successful completion
TMLIBAPP_ERR_INVALID_INSTANCE	Returned if the desired instance is not open.

Description

This function will shut down an instance of the decoder. The instance must have been stopped prior to calling the function.

tmolAdecMpegGetInstanceSetup

```
extern tmLibappErr_t tmolAdecMpegGetInstanceSetup(
   Int                              instance,
   ptmolAdecMpegInstanceSetup_t    *setup
);
```

Parameters

instance	Instance value, as returned by tmolAdecMpegOpen.
setup	Pointer to a setup structure pointer.

Return Codes

`TMLIBAPP_OK`	Returned on successful completion
`TMLIBAPP_ERR_INVALID_INSTANCE`	Returned if the desired instance is not open.

Description

The `tmolAdecMpegGetInstanceSetup` function is used to return a pointer to the decoders default OL Layer instance setup structure. The decoder creates this structure when the component is opened. After obtaining the pointer to the structure, the application can initialize specific instance values before calling `tmolAdecMpegInstanceSetup`.

tmolAdecMpegInstanceSetup

```
extern tmLibappErr_t tmolAdecMpegInstanceSetup (
   Int                             instance,
   ptmolAdecMpegInstanceSetup_t    setup
);
```

Parameters

instance	Instance value, as returned by tmalAdecMpegOpen.
setup	Pointer to the setup structure.

Return Codes

`TMLIBAPP_OK`	Returned on successful completion.
`TMLIBAPP_ERR_INVALID_INSTANCE`	Returned if the desired instance is not open.
`TMLIBAPP_ERR_NULL_PROGRESSFUNC`	Returned if the progress function callback pointer is Null.
`TMLIBAPP_ERR_NULL_DATAINFUNC`	Returned if the datain function callback pointer is Null.
`TMLIBAPP_ERR_NULL_DATAOUTFUNC`	Returned if the dataout function callback pointer is Null.
`TMLIBAPP_ERR_NULL_CONTROLFUNC`	Returned if the control function callback pointer is Null.
`TMLIBAPP_ERR_UNSUPPORTED_DATACLASS`	Returned if the input/output dataClass is not `avdcAudio`.
`TMLIBAPP_ERR_UNSUPPORTED_DATATYPE`	Returned if the input dataType is not `atfMPEG` or the output dataType is not `atfLinearPCM`.
`TMLIBAPP_ERR_UNSUPPORTED_DATASUBTYPE`	Returned if the input dataSubtype is not either `amfMPEG_Layer1`, `amfMPEG_Layer2` or `amfMPEG_Layer3`, or the output data subtype is not `apfStereo16`.

`TMLIBAPP_ERR_NULL_IODESC`	Can assert if the input descriptor is Null.
`TMLIBAPP_ERR_NO_QUEUE`	Returned if the output descriptor has no full.

Description

This function configures the decoder.

tmolAdecMpegInstanceConfig

```
extern tmLibappErr_t tmolAdecMpegInstanceConfig  (
   Int                 instance,
   UInt32              flags,
   ptsaControlArgs_t   args
);
```

Parameters

instance	Instance value, as returned by tmolAdecMpegOpen
flags	Not used.
args	Pointer to the configuration arguments.

Return Codes

`TMLIBAPP_OK`	Returned on successful completion.
`TMLIBAPP_ERR_INVALID_INSTANCE`	Returned if the desired instance is not open.
`TMLIBAPP_ERR_INVALID_COMMAND`	Returned if the configuration command is not recognized.

Description

This function can be used to change instance parameters after the component has been initialized and during data streaming operation. Right now no commands are implemented. This might change in the future.

tmolAdecMpegStart

```
extern tmLibappErr_t tmolAdecMpegStart (
   Int   instance
);
```

Parameters

instance	Instance value, as returned by tmalAdecMpegOpen.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_INVALID_INSTANCE	Returned if the desired instance is not open or setup.

Description

This function begins data streaming for the decoder. At the AL layer, it invokes a function that is an infinite while loop. At the OL layer, this while loop is spawned as a task.

tmolAdecMpegStop

```
extern tmLibappErr_t tmolAdecMpegStop (
   Int   instance
);
```

Parameters

instance	Instance value, as returned by tmxlAdecMpegOpen.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_INVALID_INSTANCE	Returned if the desired instance is not open or setup.

Description

This function stops the audio decoder from streaming data.