



08-6862-SIS-ZCH66
JUL, 25, 2008
3.0

Application Programmers Interface for Mpeg Demuxer

ABSTRACT:

Application Programmers Interface for Mpeg Demuxer

KEYWORDS:

MPEG, Program Stream, Transport Stream, Parser, Demuxer, Demultiplexer

APPROVED:

Wang Zening

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
0.1	01-July-2008	Bo Zhao	Initial Draft
	02-July-2008	Bo Zhao	Add skip control option
	14-July-2008	Bo Zhao	Make the API extensible for TS
1.0	15-July-2008	Bo Zhao	The first release
2.0	25-July-2008	Bo Zhao	Add AC3/LPCM/DTS/AAC audio support.
			Add fast scan to support seek.
2.0	08-Aug-2008	Bo Zhao	Add sequence header output
3.0	14-Aug-2008	Bo Zhao	Support MPEG2 TS

Table of Contents

1	Introduction	4
1.1	Purpose	4
1.2	Scope	4
1.3	Audience Description	4
1.4	References	4
1.4.1	Standards	4
1.4.2	Freescall Multimedia References	4
1.5	Definitions, Acronyms, and Abbreviations	5
1.6	Document Location	5
2	API Description	6
2.1	Enumeration and typedefs	6
2.2	Data structures	9
2.3	Application Programmer Interface	13
2.3.1	Call back functions	13
2.3.2	FSL_MPG_DEMUX_Open	15
2.3.3	FSL_MPG_DEMUX_GetPSI	16
2.3.4	FSL_MPG_DEMUX_Probe	16
2.3.5	FSL_MPG_DEMUX_Control	17
2.3.6	FSL_MPG_DEMUX_Process	17
2.3.7	FSL_MPG_DEMUX_Scan	18
2.3.8	FSL_MPG_DEMUX_Close	18
2.3.9	FSL_MPG_DEMUX_Version_Info	18
3	Example Lib Usage	19
4	Exception handling	20
4.1	Handling for small input buffers	20
4.2	Handling for stream errors	20
4.3	Handling for stream discontinuity	20

1 Introduction

1.1 Purpose

This document gives details of the application programmer's interface for mpeg demuxer, or named demultiplexer.

1.2 Scope

This document does not detail the implementation of the mpeg demuxer. It only explains the APIs and data structures exposed to the application developer for using the mpeg demuxer library.

1.3 Audience Description

The reader is expected to have basic understanding of MPEG1/2 system specification.

1.4 References

1.4.1 Standards

- ISO/IEC 11172-1(1993): Information technology–Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s–Part 1: Systems.
- ISO/IEC 13818-1(2000): Information technology--Generic coding of moving pictures and associated audio information: Systems.

1.4.2 Freescale Multimedia References

- Mpeg Demuxer Interface Header – mpg_demuxer_api.h, fsl_datatype.h
- Mpeg Demuxer Application Code – application.c

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
API	Application Programming Interface
ARM	Advanced RISC Machine
MPEG	Motion Picture Expert Group
PS	Program Stream
PSM	Program Stream Map
ES	Elementary Stream
PES	Packetized Elementary Stream
DTS	Decoding Time Stamp
PTS	Presentation Time Stamp
AC3	Dolby Digital
LPCM	Linear Pulse Code Modulation
DTS	Digital Theater System
AAC	Advanced Audio Coding
TS	Transport Stream
PSI	Program Specific Information
PAT	Program Association Table
PMT	Program Map Table
CAT	Conditional Access Table
FSL	Freescale
ISO	International Standards Organization
IEC	International Electrotechnical Commission
RVDS	ARM Real View Development Suite
OS	Operating System
TBD	To Be Determined

1.6 Document Location

docs/mpg2

2 API Description

This section describes the API interface of mpeg demuxer.

Supported features:

- MPEG1 system stream, MPEG2 program stream, MPEG2 PES, MPEG2 TS.
- Video: MPEG1/2.
- Audio: MPEG layer 1/2/3, AAC, AC3, LPCM, DTS.

Remarks:

The demuxer only removes the headers and outputs the ES contained. The demuxer does not guarantee the first output byte is the beginning of an ES access point.

2.1 Enumeration and typedefs

FSL_MPG_DEMUX_MEDIA_TYPE_T

This enumeration specifies the media type in streams.

```
typedef enum
{
    FSL_MPG_DEMUX_NOMEDIA=0,
    FSL_MPG_DEMUX_AUDIO_STREAM,
    FSL_MPG_DEMUX_VIDEO_STREAM,
    FSL_MPG_DEMUX_UNKNOWN_MEDIA
} FSL_MPG_DEMUX_MEDIA_TYPE_T;
```

value

FSL_MPG_DEMUX_NOMEDIA

FSL_MPG_DEMUX_AUDIO_STREAM

FSL_MPG_DEMUX_VIDEO_STREAM

FSL_MPG_DEMUX_UNKNOWN_MEDIA

meaning

No valid media found

Audio

Video

Undefined media type

FSL_MPG_DEMUX_CHANNEL_MODE_T

This enumeration specifies audio channel mode.

```
typedef enum
{
    FSL_MPEG_A_CH_MODE_STEREO=0,
    FSL_MPEG_A_CH_MODE_JOINTSTEREO,
    FSL_MPEG_A_CH_MODE_DUALCHANNEL,
    FSL_MPEG_A_CH_MODE_SINGLECHANNEL,
    FSL_MPEG_A_CH_MODE_UNKNOWN
} FSL_MPG_DEMUX_CHANNEL_MODE_T;
```

Value	Meaning
-------	---------

FSL_MPEG_A_CH_MODE_STEREO	Stereo
FSL_MPEG_A_CH_MODE_JOINTSTEREO	Joint stereo
FSL_MPEG_A_CH_MODE_DUALCHANNEL	Dual channel
FSL_MPEG_A_CH_MODE_SINGLECHANNEL	Single channel
FSL_MPEG_A_CH_MODE_UNKNOWN	Unknown

FSL_MPG_DEMUX_VIDEO_TYPE_T

This enumeration specifies video type.

```
typedef enum
{
    FSL_MPG_DEMUX_NOVIDEO = 0,
    FSL_MPG_DEMUX_MPEG2_VIDEO,
    FSL_MPG_DEMUX_UNKNOWN_VIDEO
} FSL_MPG_DEMUX_VIDEO_TYPE_T;
```

Value	Meaning
FSL_MPG_DEMUX_NOVIDEO	No video
FSL_MPG_DEMUX_MPEG2_VIDEO	MPEG1/2 video
FSL_MPG_DEMUX_UNKNOWN_VIDEO	Unknown

FSL_MPG_DEMUX_AUDIO_TYPE_T

This enumeration specifies audio type.

```
typedef enum
{
    FSL_MPG_DEMUX_NOAUDIO = 0,
    FSL_MPG_DEMUX_MP1_AUDIO,
    FSL_MPG_DEMUX_MP2_AUDIO,
    FSL_MPG_DEMUX_MP3_AUDIO,
    FSL_MPG_DEMUX_AAC_AUDIO,
    FSL_MPG_DEMUX_AC3_AUDIO,
    FSL_MPG_DEMUX_DTS_AUDIO,
    FSL_MPG_DEMUX_PCM_AUDIO,
    FSL_MPG_DEMUX_UNKNOWN_AUDIO
} FSL_MPG_DEMUX_AUDIO_TYPE_T;
```

Value	Meaning
FSL_MPG_DEMUX_NOAUDIO	No audio
FSL_MPG_DEMUX_MP1_AUDIO	MPEG layer 1 audio
FSL_MPG_DEMUX_MP2_AUDIO	MPEG layer 2 audio
FSL_MPG_DEMUX_MP3_AUDIO	MPEG layer 3 audio(MP3)
FSL_MPG_DEMUX_AAC_AUDIO	MPEG2 AAC audio
FSL_MPG_DEMUX_AC3_AUDIO	AC3 audio
FSL_MPG_DEMUX_DTS_AUDIO	DTS audio
FSL_MPG_DEMUX_PCM_AUDIO	LPCM audio
FSL_MPG_DEMUX_UNKNOWN_AUDIO	Unknown

FSL_MPG_DEMUX_CMD_T

This enumeration specifies the control command.

```
typedef enum
{
    FSL_MPG_DEMUX_CMD_RESET=0,
    FSL_MPG_DEMUX_CMD_RESYNC,
    FSL_MPG_DEMUX_CMD_SELECT_PROGRAM,
    FSL_MPG_DEMUX_CMD_ENABLE_STREAM,
    FSL_MPG_DEMUX_CMD_DISABLE_STREAM,
    FSL_MPG_DEMUX_CMD_SET_VBUFSIZE,
    FSL_MPG_DEMUX_CMD_SET_ABUFSIZE,
    FSL_MPG_DEMUX_CMD_SET_SKIPFLAG,
    FSL_MPG_DEMUX_CMD_GET_MAXPESSIZE_AUDIO,
    FSL_MPG_DEMUX_CMD_GET_MAXPESSIZE_VIDEO,
    FSL_MPG_DEMUX_CMD_GET_MPEGSH_PTR,
    FSL_MPG_DEMUX_CMD_GET_MPEGSH_SIZE,
    FSL_MPG_DEMUX_CMD_MAX
} FSL_MPG_DEMUX_CMD_T;
```

Command	Comments
FSL_MPG_DEMUX_CMD_RESET	Reset all the context of the demuxer
FSL_MPG_DEMUX_CMD_RESYNC	Reset the input stream context
FSL_MPG_DEMUX_CMD_SELECT_PROGRAM	Select one program in TS(channel)
FSL_MPG_DEMUX_CMD_ENABLE_STREAM	Select one stream with stream ID
FSL_MPG_DEMUX_CMD_DISABLE_STREAM	Unselect one stream with stream ID
FSL_MPG_DEMUX_CMD_SET_VBUFSIZE	Set the maximum video buffer size the application can provide.
FSL_MPG_DEMUX_CMD_SET_ABUFSIZE	Set the maximum audio buffer size the application can provide.
FSL_MPG_DEMUX_CMD_SET_SKIPFLAG	Set 1 to enable skip one PES when request output ES buffer failed, otherwise just return without skipping operation.
FSL_MPG_DEMUX_CMD_GET_MAXPESSIZE_AUDIO	Get the maximum audio PES size the demuxer scanned.
FSL_MPG_DEMUX_CMD_GET_MAXPESSIZE_VIDEO	Get the maximum video PES size the demuxer scanned.
FSL_MPG_DEMUX_CMD_GET_MPEGSH_PTR	Get MPEG sequence header buffer pointer
FSL_MPG_DEMUX_CMD_GET_MPEGSH_SIZE	Get MPEG sequence header buffer size
FSL_MPG_DEMUX_CMD_MAX	Reserved

FSL_MPG_DEMUX_RET_TYPE_T

This enumeration defines the return code of the demuxer.

```
typedef enum
{
    FSL_MPG_DEMUX_RT_SUCCESS=0,
    FSL_MPG_DEMUX_RT_MOREDATA,
    FSL_MPG_DEMUX_RT_NVALID_STREAM,
    FSL_MPG_DEMUX_RT_EOS,
}
```



```

FSL_MPG_DEMUX_RT_NVALIDCMD,
FSL_MPG_DEMUX_RT_NULL_PARSER,
FSL_MPG_DEMUX_RT_MALLOC_FAILED,
FSL_MPG_DEMUX_RT_DISCONTINUITY_AUDIO,
FSL_MPG_DEMUX_RT_DISCONTINUITY_VIDEO,
FSL_MPG_DEMUX_RT_INPUTBUF_SMALL,
FSL_MPG_DEMUX_RT_VITAL_FAULT
} FSL_MPG_DEMUX_RET_TYPE_T;

```

Return code	Comments
FSL_MPG_DEMUX_RT_SUCCESS	Success
FSL_MPG_DEMUX_RT_MOREDATA	More data needed to be fed to steam buffers
FSL_MPG_DEMUX_RT_NVALID_STREAM	Not supported stream or error found
FSL_MPG_DEMUX_RT_EOS	End of stream found
FSL_MPG_DEMUX_RT_NVALIDCMD	The control command is not supported
FSL_MPG_DEMUX_RT_NULL_PARSER	Passed demuxer instance is NULL
FSL_MPG_DEMUX_RT_MALLOC_FAILED	Memory allocation through call back failed
FSL_MPG_DEMUX_RT_DISCONTINUITY_AUDIO	When request audio output buffer failed, the demuxer will skip one PES and return
FSL_MPG_DEMUX_RT_DISCONTINUITY_VIDEO	When request video output buffer failed, the demuxer will skip one PES and return
FSL_MPG_DEMUX_RT_INPUTBUF_SMALL	The input buffer size is smaller than 188*2.
FSL_MPG_DEMUX_RT_VITAL_FAULT	Unknown fatal error

2.2 Data structures

FSL_MPG_DEMUX_BD_T

This structure specifies the input stream buffer descriptor.

```

typedef struct FSL_MPG_DEMUX_BD_S
{
    VOID *    pPtr;
    VOID *    pCnxt;
    U32       uliLen;
    U32       uliFill;
} FSL_MPG_DEMUX_BD_T;

```

Description of structure *FSL_MPG_DEMUX_BD_T*

pPtr

The buffer pointer (start address of the buffer).

pCnxt

Pointer to versatile context.

uliLen

Buffer size.

uliFill

Filled buffer length from the start of the buffer (valid buffer size).

FSL_MPG_DEMUX_BD_SET_T

This structure specifies input stream buffer set.

```
typedef struct FSL_MPG_DEMUX_BD_SET_S
{
    FSL_MPG_DEMUX_BD_T *    pPtr;
    U16                      usiCntBDSet;
} FSL_MPG_DEMUX_BD_SET_T;
```

Description of structure *FSL_MPG_DEMUX_BD_SET_T*

pPtr

Pointer to input buffer descriptors.

usiCntBDSet

The total number of input buffer.

FSL_VIDEO_PROPERTY_T

This structure specifies the video property.

```
typedef struct FSL_VIDEO_PROPERTY_S
{
    FSL_MPG_DEMUX_VIDEO_TYPE_T    enuVideoType;
    U32                            uliVideoID;
    U32                            uliVideoWidth;
    U32                            uliVideoHeight;
    U32                            uliVideoBitRate;
    U32                            uliFRNumerator;
    U32                            uliFRDenominator;
} FSL_VIDEO_PROPERTY_T;
```

Description of structure *FSL_VIDEO_PROPERTY_T*

enuVideoType

Video type.

uliVideoID

Stream ID.

uliVideoWidth

Video horizontal size in pixels.

uliVideoHeight

Video vertical size in pixels.

uliVideoBitRate

Video bitrate **in unit of bits/s**.

uliFRNumerator

Video frame rate numerator.

uliFRDenominator

Video frame rate denominator.

FSL_AUDIO_PROPERTY_T

This structure specifies the audio property.

```
typedef struct FSL_AUDIO_PROPERTY S
{
    FSL_MPG_DEMUX_AUDIO_TYPE T    enuAudioType;
    U32                            uliAudioID;
    U32                            uliAudioSampleRate;
    U16                            usiAudioChannels;
    FSL_MPG_DEMUX_CHANNEL_MODE T  enuAudioChannelMode;
    U32                            uliAudioBitRate;
} FSL_AUDIO_PROPERTY_T;
```

Description of structure *FSL_AUDIO_PROPERTY_T*

enuAudioType

Audio type.

uliAudioID

Stream ID of this audio stream.

uliAudioSampleRate

Audio sample rate in unit of Hz.

usiAudioChannels

Audio channels number.

enuAudioChannelMode

Audio channel mode.

uliAudioBitRate

Audio bitrate in unit of 1000 bits/s .

FSL_MPEGSTREAM_T

This structure specifies the stream property.

```
typedef struct FSL_MPEGSTREAM S
{
    FSL_MPG_DEMUX_MEDIA_TYPE T    enuStreamType;
    U32                            uliPropertyValid;
    union
    {
        FSL_VIDEO_PROPERTY T      VideoProperty;
        FSL_AUDIO_PROPERTY T      AudioProperty;
    } MediaProperty;
} FSL_MPEGSTREAM_T;
```

Description of structure *FSL_MPEGSTREAM_T*

enuStreamType

Media type, video or audio.

uliPropertyValid

This property is valid or not.

MediaProperty

Video or audio property.

FSL_MPG_DEMUX_SYSINFO_T

This structure specifies the system information.

```
typedef struct FSL_MPG_DEMUX_SYSINFO_S
{
    U32                uliNoStreams;
    U32                uliPSFlag;
    FSL_MPEGSTREAM_T   Stream[NO_MEDIA_SUPPORT_MAX];
} FSL_MPG_DEMUX_SYSINFO_T;
```

Description of structure *FSL_MPG_DEMUX_SYSINFO_T*

uliNoStreams

Number of streams found in targeted stream.

uliPSFlag

The stream is valid program stream or not.

Stream[]

Stream property of the found streams.

FSL_MPG_DEMUX_PSI_T

This structure specifies the program information.

```
typedef struct FSL_MPG_DEMUX_PSI_S
{
    U8    IsTS;
    U8    Programs;
    U16    ProgramNumber[NO_PROGRAM_SUPPORT_MAX];
} FSL_MPG_DEMUX_PSI_T;
```

Description of structure *FSL_MPG_DEMUX_PSI_T*

IsTS

Set to 1 if the stream is TS.

Programs

The number of programs indicated in PAT.

ProgramNumber[]

The program number array in PAT.

FSL_MPG_DEMUX_T

This structure specifies the demuxer instance.

```
typedef struct FSL_MPG_DEMUX_S
{
    FSL_MPG_DEMUX_PSI_T   TS_PSI;
    FSL_MPG_DEMUX_SYSINFO_T   SystemInfo;
    FSL_MPG_DEMUX_BD_SET_T   InputBuffer;
    VOID *                  pDemuxContext;
    VOID *                  pAppContext;
    /*call back functions */
}
```

```

    VOID *      (*DemuxMalloc) (VOID * pAppContext, U32 Size);
    VOID      (*DemuxFree) (VOID * pAppContext, VOID *Ptr );
    VOID *      (*DemuxRequestBuffer) (VOID * pAppContext, U32
        StreamID, U32 Size, U32 NewPacket);
    S32      (*DemuxOutput) (VOID * pAppContext, U32 StreamID, VOID
        *pES, U32 Size, U64 PTS, U64 DTS, U32 Flag);
    S32      (*DemuxOutputIndex) (FSL VOID * pAppContext, U32
        StreamID, U32 BufIndex, S32 BufOffset, U64 PTS, U64
        DTS, U32 Flag);
} FSL MPG DEMUX T;

```

Description of structure *FSL_MPG_DEMUX_T*

TS_PSI

PAT information in TS.

SystemInfo

Parsed system information.

InputBuffer

Input stream buffer sets.

pDemuxContext

Pointer to the demuxer context.

pAppContext

Pointer to application context.

DemuxMalloc

Call back function for memory allocation.

DemuxFree

Call back function for memory free.

DemuxRequestBuffer

Call back function for output buffer request.

DemuxOutput

Call back function for pass out ES buffer.

DemuxOutputIndex

Call back function for pass out time stamps with buffer index and offset.

The details of the call back functions please see following section.

2.3 Application Programmer Interface

2.3.1 Call back functions

2.3.1.1 DemuxMalloc

Memory allocation.

C prototype:

```
VOID * (*DemuxMalloc) (VOID * pAppContext, U32 Size);
```

Arguments:

- pAppContext Pointer to application context;
- Size The size of requesting memory in bytes.

Return value:

Pointer to the allocated memory.

2.3.1.2 DemuxFree

Memory free.

C prototype:

```
VOID (*DemuxFree) (VOID * pAppContext, VOID *Ptr);
```

Arguments:

- pAppContext Pointer to application context;
- Ptr Pointer to the memory to be freed.

Return value:

None.

2.3.1.3 DemuxRequestBuffer

Request output buffer from application.

C prototype:

```
VOID * (*DemuxRequestBuffer) (VOID * pAppContext, U32 StreamID,  
                                  U32 Size, U32 NewPacket );
```

Arguments:

- pAppContext Pointer to application context;
- StreamID The ID of the stream to be passed out;
- Size The size of the requesting ES buffer in bytes;
- NewPacket Set to 1 to indicate the requested buffer is for a new ES packet.

Return value:

Pointer to the requested ES buffer.

2.3.1.4 DemuxOutput**C prototype:**

```
S32 (*DemuxOutput) (VOID * pAppContext, U32 StreamID, VOID *pES,  
                                  U32 Size, U64 PTS, U64 DTS, U32 Flag);
```

Arguments:

- pAppContext Pointer to application context;
- StreamID The ID of the stream the ES buffer associated;
- pES Pointer to the ES buffer passing out;
- Size The valid ES stream in the buffer;
- PTS Presentation time stamp(33 bits), in units of 90KHz;
- DTS Decoding time stamp(33 bits), in units of 90KHz;
- Flag Versatile flag. See below.

Bit	meaning
0	PTS valid flag
1	DTS valid flag
2	New PES flag. 1 means this is a new PES, 0 means the buffer is the rest of a PES.
others	Reserved

Return value:

- 0 Success pass out.
 -1 Failed because the stream ID is not selected.

2.3.1.5 DemuxOutputIndex

C prototype:

```
S32 (*DemuxOutputIndex) (FSL_VOID * pAppContext, U32 StreamID,
                        U32 BufIndex, S32 BufOffset, U64 PTS, U64 DTS, U32 Flag);
```

Arguments:

- pAppContext Pointer to application context;
- StreamID The ID the ES associated;
- BufIndex The index of the input buffer which contains the new PES;
- BufOffset The offset of the first byte of PES in the input buffer;
- PTS Presentation time stamp(33 bits), in units of 90KHz;
- DTS Decoding time stamp(33 bits), in units of 90KHz;
- Flag Versatile flag. See below.

Bit	meaning
0	PTS valid flag
1	DTS valid flag
others	Reserved

Return value:

- 0 Success pass out.
 -1 Failed because the stream ID is not selected (valid).

2.3.2 FSL_MPG_DEMUX_Open

Open a demuxer instance.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_Open(FSL_MPG_DEMUX_T *pDemuxer);
```

Arguments:

- pDemuxer Pointer to demuxer instance.

Return value:

- FSL_MPG_DEMUX_RT_SUCCESS Parser open success;

- FSL_MPG_DEMUX_RT_NULL_PARSER Passed demuxer object is NULL;
- FSL_MPG_DEMUX_RT_MALLOC_FAILED Failed to allocate memory for internal context.

Return code	Comments
FSL_MPG_DEMUX_RT_SUCCESS	Success
FSL_MPG_DEMUX_RT_MOREDATA	More data needed to be fed to steam buffers
FSL_MPG_DEMUX_RT_NVALID_STREAM	Not supported stream or error found
FSL_MPG_DEMUX_RT_EOS	End of stream found
FSL_MPG_DEMUX_RT_NVALIDCMD	The control command is not supported
FSL_MPG_DEMUX_RT_NULL_PARSER	Passed demuxer instance is NULL
FSL_MPG_DEMUX_RT_MALLOC_FAILED	Memory allocation through call back failed
FSL_MPG_DEMUX_RT_DISCONTINUITY	When request output buffer failed, the demuxer will skip one PES and return
FSL_MPG_DEMUX_RT_VITAL_FAULT	Unknown fatal error

2.3.3 FSL_MPG_DEMUX_GetPSI

Scan the stream to get PSI information.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_GetPSI(FSL_MPG_DEMUX_T *pDemuxer);
```

Arguments:

- pDemuxer Pointer to demuxer object.

Return value:

- FSL_MPG_DEMUX_RT_SUCCESS PSI information successfully detected.
- FSL_MPG_DEMUX_RT_MOREDATA More data shall be fed;
- FSL_MPG_DEMUX_RT_NVALID_STREAM Fatal Error found in stream, not supported stream .
- FSL_MPG_DEMUX_RT_INPUTBUF_SMALL The data size in input buffer is smaller than 188*2, which is the least stream needed to detect sync code.
- FSL_MPG_DEMUX_RT_MALLOC_FAILED Malloc failed.

2.3.4 FSL_MPG_DEMUX_Probe

Scan the stream to get system information.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_Probe(FSL_MPG_DEMUX_T *pDemuxer);
```

Arguments:

- pDemuxer Pointer to demuxer object.

Return value:

- FSL_MPG_DEMUX_RT_SUCCESS System information successfully got. When the stream contains system header, the demuxer can get the maximum number of video and audio streams.

So the demuxer return success when all the media information is got.

- FSL_MPG_DEMUX_RT_MOREDATA More data shall be fed;
- FSL_MPG_DEMUX_RT_NVALID_STREAM Fatal Error found in stream, not supported stream .

2.3.5 FSL_MPG_DEMUX_Control

Issue control command.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_Control(FSL_MPG_DEMUX_T
 *pDemuxer, FSL_MPG_DEMUX_CMD_T Cmd, VOID *Args);
```

Arguments:

- pDemuxer Pointer to demuxer object;
- Cmd Command. Refer definition of FSL_MPG_DEMUX_CMD_T;
- Args Pointer to argument. When set parameters, the parameters are passed by Args. When get information, the information is passed out by Args.

Return value:

- FSL_MPG_DEMUX_RT_SUCCESS Control success;
- FSL_MPG_DEMUX_RT_NVALIDCMD The command is not supported or the arguments are unvalid;
- FSL_MPG_DEMUX_RT_NULL_PARSER Passed demuxer is NULL.

2.3.6 FSL_MPG_DEMUX_Process

Parse the stream in the input buffers.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_Process(FSL_MPG_DEMUX_T *pDemuxer);
```

Arguments:

- pDemuxer Pointer to demuxer object.

Return value:

- FSL_MPG_DEMUX_RT_MOREDATA More data shall be fed into input buffers;
- FSL_MPG_DEMUX_RT_NVALID_STREAM Error found in stream;
- FSL_MPG_DEMUX_RT_EOS Processed to the end of stream;
- FSL_MPG_DEMUX_RT_VITAL_FAULT Vital error: the stream is not probed or ES pass out failed;
- FSL_MPG_DEMUX_RT_DISCONTINUITY_AUDIO Request audio output buffer failed, the demuxer skipped one PES and return if skip is enabled. If skip is disabled the demuxer just return.
- FSL_MPG_DEMUX_RT_DISCONTINUITY_VIDEO Request video output buffer failed, the demuxer skipped one PES and return if skip is enabled. If skip is disabled the demuxer just return.

2.3.7 FSL_MPG_DEMUX_Scan

Fast scan the stream in input buffers, output time stamp and offset through callback function.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_Scan(FSL_MPG_DEMUX_T *pDemuxer);
```

Arguments:

- pDemuxer Pointer to demuxer object.

Return value:

- FSL_MPG_DEMUX_RT_MOREDATA More data shall be fed into input buffers;
- FSL_MPG_DEMUX_RT_NVALID_STREAM Error found in stream;
- FSL_MPG_DEMUX_RT_EOS Processed to the end of stream;
- FSL_MPG_DEMUX_RT_VITAL_FAULT Vital error: the stream is not probed or pass out failed.

2.3.8 FSL_MPG_DEMUX_Close

Close the demuxer instance.

C prototype:

```
FSL_MPG_DEMUX_RET_TYPE_T FSL_MPG_DEMUX_Close(FSL_MPG_DEMUX_T *pDemuxer);
```

Arguments:

- pDemuxer Pointer to demuxer object.

Return value:

- FSL_MPG_DEMUX_RT_SUCCESS Success;
- FSL_MPG_DEMUX_RT_NULL_PARSER Passed demuxer is NULL.

2.3.9 FSL_MPG_DEMUX_Version_Info

Get version information.

C prototype:

```
const U8 * FSL_MPG_DEMUX_Version_Info(VOID);
```

Arguments:

None.

Return value:

String of version information.

3 Example Lib Usage

Please refer to application.c.

4 Exception handling

In embedded systems, the memory resource is limited. But the PES size can be as large as 64K bytes in theory. In another hand, there may exist bit errors in the input streams. All these may cause exceptions. The exceptions resulted shall be handled by the cooperation of the application and the demuxer.

4.1 Handling for small input buffers

The demuxer assumes the input buffer can hold one complete PES packet. During the probe or process steps, the PES size may be larger than the total size of the input buffers. It is for the application to find this exception and take some actions.

Exception occurs when:

- 1) Parser returned FSL_MPG_DEMUX_RT_MOREDATA;
- 2) All the input buffers are valid.

Handling procedure:

- 1) Issue FSL_MPG_DEMUX_CMD_RESYNC command to demuxer;
- 2) Release all the input buffers;
- 3) Continue feeding stream into input buffer;
- 4) Continue probing or processing.

4.2 Handling for stream errors

When the demuxer finds fatal errors in stream, it returns with FSL_MPG_DEMUX_RT_NVALID_STREAM. It is up to the application to handle this exception. The application can either abort this stream or go ahead as the above section before skipping the error part.

4.3 Handling for stream discontinuity

When demuxer fails to request output audio/video ES buffer, it returns with FSL_MPG_DEMUX_RT_DISCONTINUITY_AUDIO/VIDEO. In this case, there are 2 options for the application to choose:

Option 1:

- 1) Allocate more ES buffers or Release some ES buffer;
- 2) Resume the parsing process.

If command `FSL_MPG_DEMUX_CMD_SET_SKIPFLAG` has been issued with 0 or has not been issued at all, the demuxer does not skip one PES, so there shall no ES discontinuity when parsing is resumed.

If command `FSL_MPG_DEMUX_CMD_SET_SKIPFLAG` has been issued with 1, discontinuity occurs in ES as the demuxer skipped one PES.

Option 2:

- 1) Ensure `FSL_MPG_DEMUX_CMD_SET_SKIPFLAG` has been issued with 1;
- 2) Resume the parsing process without buffer adaptation.

In this case there must be discontinuity in ES.

In both options, the application shall handle the A/V sync issue if discontinuity occurs.