MP4 Parser library API

Version: 1.2

Date: 03/19/2009

Freescale Semiconductor, Shanghai Software Development Center
UnitA, 20F, 560#, SongTao Road, Pu Dong New District
Shanghai, 201203, P.R of China

Revision History

Version	Date	Revised By	Description of Changes
1.0	08/20/2008	Amanda Lin	Initial version.
1.1	11/13/08	Amanda Lin	Update for MP4 parser lib REV5.12.4
1.2	03/19/2009	Peng Du	Update for MP4 parser lib REV5.13.0

1. API functions

1.1. Initialization

```
MP4Err MP4ParserModuleInit (sMP4ParserObjectType  *ObjectType,
                            FILE       *fp,
                            u32        *total_tracks,
                            void        *demuxer,
                            sFunctionPtrTable     *ptrFuncPtrTable);
```

Description:
initialise all the basic data structures and necessary variables. This is the first function to call.
Arguments:
sMP4ParserObjectType * ObjectType
IN/OUT, internal data structure of MP4 parser library. Please refer to section "data structures"- sMP4ParserObjectType.

FILE *fp ,
IN, file pointer of the MP4 file. Not used now and set to NULL pointer.

u32 *total_tracks
OUT, to get the number to total tracks in this MP4 file. The tracks include all the A/V tracks and non-A/V tracks (eg. Indicator tracks, etc)
.
void *demuxer
IN, the application context for the callback functions. The MP4 parser library never change its value and uses it as an argument of all file callback functions.

sFunctionPtrTable *ptrFuncPtrTable
IN, pointer to callback function table. Please refer to section "data structures"- sFunctionPtrTable.

Return value:
MP4NoErr – success
Other error codes - Failure

1.2. Get file information

```
MP4Err MP4ParserGetFileInfo (sMP4ParserObjectType *ObjectType,
                             sMP4ParserFileInfo   *FileInfo);
```

Description:
Get file information.

Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object, initialized by "MP4ParserModuleInit".

sMP4ParserFileInfo *FileInfo
OUT, file information.

Return value:
MP4NoErr – success
Other error codes - Failure

1.3. Get stream information

```
MP4Err MP4ParserGetStreamInfo (  sMP4ParserObjectType *ObjectType,
                                 sMP4ParserFileInfo *FileObjectType,
                                 u32 track_number,
                                 sMP4ParserReadByteInfo * ReadInfo);
```

Description:
Get detailed stream information.

Arguments:
sMP4ParserObjectType *ObjectType
IN/OUT, MP4 library object, initialized by "MP4ParserModuleInit".
After this function is called, detailed information can be got from this structure.

Baisc track type: ObjectType ->trak[track index]->decodertype
There are three basic track types
MP4_VIDEO - video track
MP4_AUDIO - audio track
MP4_HINT – hint track

Codec type of a track:
MP4PmObjectType->trak[m_iTrackIndex]->decodertype
please refer to section "data structures"- eCodecType.

Media duration in nano-seconds:
MP4PmObjectType->media_duration

And detailed stream properties can be got from the member arrary:
"MP4PmObjectType->media[MP4_PARSER_MAX_STREAM]".
For the array definition, please refer to section "data structures"- MP4Media.

sMP4ParserFileInfo *FileInfo
IN/OUT, file information, initialized by “MP4ParserGetFileInfo”. And this function will fill detailed stream information in this data structure.

u32 track_number,
IN, number of tracks, including both A/V and non-A/V tracks, got from “MP4ParserModuleInit”.

sMP4ParserReadByteInfo * ReadInfo
IN, internal reading information of MP4 parser library.

Return value:
MP4NoErr – success
Other error codes - Failure

1.4. Allocate working buffers

```
MP4Err MP4ParserAllocateMemory(sMP4ParserObjectType *ObjectType,
                               sMP4ParserReadByteInfo *ReadByteInfo,
                               sMP4ParserFileInfo     * FileInfo,
                               u32 total_tracks);
```

Description:
Allocate internal working buffers of MP4 parser library.
MP4 parser has one internal working buffer for each stream. It’s large enough to hold the biggest access unit of this stream. When reading a stream, the access unit is read from the file to this buffer, and the caller can get data from this buffer.

And for some H.264 streams with NAL length field of 2 bytes, the video AU (access units) need to be resembled to insert NAL start code before each NAL data and the output AU size will change. This will need another working buffer of about the same size. And if such H.264 streams are present, this extra working buffer is also allocated here.

After this function is called, the MP4 parser library is ready to read stream data.

Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object, initialized by “MP4ParserModuleInit”.

sMP4ParserReadByteInfo * ReadInfo
IN, internal reading information of MP4 parser library.

sMP4ParserFileInfo *FileInfo
IN, file information, initialized by “MP4ParserGetFileInfo”.

u32 total_tracks,
IN, number of tracks, including both A/V and non-A/V tracks, got from "MP4ParserModuleInit".

Return value:
MP4NoErr – success
Other error codes - Failure

1.5. Read stream data

MP4Err MP4ParserReadFile (sMP4ParserObjectType *ObjectType,
sMP4ParserReadByteInfo *ReadInfo,
u32 track_count);

Description:
Read the next sample (access unit) of a specified track.

Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object, initialized by "MP4ParserModuleInit".

sMP4ParserReadByteInfo * ReadInfo
IN/OUT, internal reading information of MP4 parser library.
If reading is successful, both the sample data and the sample size can be got from this data structure:
Sample data -
ReadInfo->track_read_info[m_iTrackIndex].MaxBufferRequired
Sample size –
MP4PmReadInfo->track_read_info[m_iTrackIndex].numBytesRead
Sample's sync flag -
ReadInfo ->track_read_info[m_iTrackIndex].sync

u32 track_count,
IN, index(0-based) of the track to read.

Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes - Failure

1.6. Free atoms

void MP4ParserFreeAtom(sMP4ParserObjectType * ObjectType)

Description:
After reading is done, MP4 parser library can be released.
This function release atom structures of MP4 library.

Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object, initialized by “MP4ParserModuleInit”.

1.7. Free internal buffers

void MP4ParserFreeMemory(sMP4ParserObjectType * MP4PmObjectType,
sMP4ParserFileInfo * MP4PmFileInfo,
sMP4ParserReadByteInfo * ReadByteInfo,
sMP4ParserUdtaList * MP4PmUserData,
u32 total_tracks)

Description:
And at last, this function free the internal non-atom working buffers of MP4 parser library. This is the last API to call for MP4 parser lib.

Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.

sMP4ParserFileInfo * MP4PmFileInfo
IN, file information

sMP4ParserReadByteInfo * ReadInfo
IN, internal reading information of MP4 parser library.

sMP4ParserUdtaList * MP4PmUserData
IN, user data list got from MP4 parser library.

u32 total_tracks
IN, number of tracks, including both A/V and non-A/V tracks, got from “MP4ParserModuleInit”. To avoid memory leak, it’s shall be the actual track number or MP4_PARSER_MAX_STREAM.

1.8. Skip video key frame

MP4Err MP4ParserSkipKeyFrame (sMP4ParserObjectType *ObjectType,
u32 track_index,
s32 forward,
s32 *eos,
s64 *seek_time_ms,
u32 *sample_number);

Description:
Call this function to skip one video key frame forward or backward.

Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.
u32 track_index
IN, index(0-based) of the track, must be video track.
s32 forward
IN, skip direction,
s32 *eos
OUT, end of stream flag
s64 *seek_time_ms
IN, target seeking time in ms.
OUT, the accurate time of the matched sample.
u32 *sample_number
OUT, number of the matched sample
Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes - Failure

1.9. Check track is indepenent or dependent on other tracks

MP4Err IsIndependentTrack(sMP4ParserObjectType *ObjectType,
u32 trak_indx,
u32* fIndpendent);
Description:
A bsac stream may be divided into several tracks. We must find independent one as entry track. This API is added for bsac to check the independence of tracks.
Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.
u32 trak_index
IN, index(0-based) of the track
u32 *fIndpendent
OUT, independent flag

Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes - Failure

1.10. Read bsac stream data

MP4Err MP4ParserReadBSAC(sMP4ParserObjectType *ObjectType,
sMP4ParserReadByteInfo *ReadInfo,
u8* outBuffer,
u32* outSize, /* size of one frame */

u32 trak_indx);

Description:
Read the next sample (access unit) of a bsac stream.
Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.
sMP4ParserReadByteInfo * ReadInfo
IN/OUT, internal reading information of MP4 parser library.
u8 *outBuffer
IN, output sample data pointer
u32 *outSize
OUT, sample size
u32 trak_indx
IN, track index
Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes - Failure

1.11. Seek bsac stream in terms of time (in ms)

MP4Err MP4ParserSeekBSAC(sMP4ParserObjectType *ObjectType,
u32 trak_indx,
s64 *seek_time_ms,
u32 *sample_number);

Description:
Function to seek bsac stream according to the passed target time in ms
Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.
u32 trak_index
IN, index(0-based) of the track
s64 *seek_time_ms
IN, target seeking time in ms.
OUT, the accurate time of the matched sample.
u32 *sample_number
OUT, number of the matched sample

Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes – Failure

1.12. Seek only one stream in terms of time (in ms)

```
MP4Err   MP4ParserSeekTrack (sMP4ParserObjectType   *ObjectType,
                             u32  trak_indx,
                             s64 *seek_time_ms,
                             u32 *sample_number);
```

Description:
Function to seek only one track stream according to the passed target time in ms
Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.
u32 trak_index
IN, index(0-based) of the track
s64 *seek_time_ms
IN, target seeking time in ms.
OUT, the accurate time of the matched sample.
u32 *sample_number
OUT, number of the matched sample

Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes – Failure

1.13. Seek all streams in terms of time (in ms)

```
MP4Err   MP4ParserSeekFile (sMP4ParserObjectType   *ObjectType,
                            u32  trak_indx,
                            s64 *seek_time_ms,
                            u32 *sample_number);
```

Description:
Function to seek the stream according to the passed target time in seconds (NOT number of bytes). And set the Reader postion.
Arguments:
sMP4ParserObjectType *ObjectType
IN, MP4 library object.
u32 trak_index
IN, index(0-based) of the track
s64 *seek_time_ms
IN, target seeking time in ms.
OUT, the accurate time of the matched sample.
u32 *sample_number
OUT, number of the matched sample
Return value:
MP4NoErr – success
MP4EOF – End of Stream. No samples can be read from this track/stream.
Other error codes – Failure

2. Error Codes

error code	value	description
MP4HasRootOD	2	support for OD, returned by MP4GetInline ... and MP4GetProfiles...
MP4EOF	1	End of File/End of Stream
MP4NoErr	0	success
MP4FileNotFoundErr	-1	File does not exist
MP4BadParamErr	-2	bad parameter
MP4NoMemoryErr	-3	short of memory
MP4IOErr	-4	IO error
MP4NoLargeAtomSupportErr	-5	Large atom is found but not supported
MP4BadDataErr	-6	bad data found
MP4VersionNotSupportedErr	-7	The spec version is not supported
MP4InvalidMediaErr	-8	Media information is invalid
MP4DataEntryTypeNotSupportedErr	-100	The data type is not supported
MP4NoQTAtomErr	-500	Not all necessary MP4 atoms are found.
MP4NotImplementedErr	-1000	Certain feature is not supported

3. Data Structures

3.1. sMP4ParserObjectType

```
typedef struct
{
   u8  *FileAddress;

   u32  streamType;
   u32  objectTypeIndication;
   u32  decoderBufferSize;
   u32  upStream;
   u32  maxStream;
   u32  avgBitStream;
   u32  NumberBytes_ADTS;
   u32  TotalFrames[MP4_PARSER_MAX_STREAM];
   u32  OffsetByte;
   u32  SetFlag;
   u32  TotalBytes[MP4_PARSER_MAX_STREAM];
   u32  OffsetUDTA;
   u32  SizeUdta;
        u32  totalstreams;   /* number of A/V streams. Other type of streams are
overlooked */
```

```
        u32  tarck_index;   /* Index of the target track to read, 0-based. Only A/V tracks
are considered. */
        u32  audio_offset;
        u32  video_offset;

   /* Variables used for the ADTS headers. */
   u32  syncword;
   u32  id;
   u32  layer;
   u32  protection_abs;
   u32  profile;
   u32  sampl_freq_idx;
   u32  private_bit;
   u32  channel_config;
   u32  original_copy;
   u32  home;
   u32  copyR_id_bit;
   u32  copyR_id_start;
   u32  frame_length;
   u32  buff_fullness;
   u32  num_of_rdb;
   u32  crc_check;
   u32  rdb_position;
   u32  crc_check_rdb;
   u32  media_time_sacle[MP4_PARSER_MAX_STREAM];
   u32  chunk_offset[MP4_PARSER_MAX_STREAM];
                /* Offset, in bytes,
                     of the 1st chunk of a track to read,
                     when file is just opened or after a seeking is performed.
                     Since seeking is peformed only on A/V tracks,
                so this chunk offset of non-A/V tracks does not change after seeking */

  u64  media_duration; /* media duration, in ns */

   MP4Movie       moov;
   MP4Track       trak[MP4_PARSER_MAX_STREAM];
   MP4Media       media[MP4_PARSER_MAX_STREAM];
   MP4TrackReader  reader[MP4_PARSER_MAX_STREAM];
   MP4Handle      decoderSpecificInfoH[MP4_PARSER_MAX_STREAM];
                /* codec specific information */
   MP4Handle      sampleH[MP4_PARSER_MAX_STREAM];

} sMP4ParserObjectType;
```

3.2. sFunctionPtrTable

```
typedef struct
{
   void  (*MP4LocalSeekFile)(FILE *, int, int, void *);
   u32   (*MP4LocalReadFile)(void *, u32, u32, FILE *, void *);
   LONGLONG   (*MP4LocalGetCurrentFilePos)(FILE *, void *);
   FILE* (*MP4LocalFileOpen)(const u8 *, const u8 *);
   LONGLONG   (*MP4LocalFileSize)(FILE *, void *);
   u32   (*MP4LocalFileClose)(FILE *);
   u32   (*MP4LocalWriteInFile)(const void *, u32, u32, FILE *);

   void  (*MP4LocalFree)(void *);
   void* (*MP4LocalCalloc)(u32, u32);
   void* (*MP4LocalMalloc)(u32);
   void* (*MP4LocalReAlloc)(void *, u32);

}sFunctionPtrTable;
```

DESCRIPTION: the callback function table that implements memory and file operations. Please refer to section “Callback functions”.

3.3. sMP4ParserFileInfo

```
typedef struct
{
        u32 NumberOfStreams; /* Number of stream in file. */
       sMP4ParserStreamInfo    MP4PmStreamArray[MP4_PARSER_MAX_STREAM];
} sMP4ParserFileInfo;
```

3.4. MP4Media

```
struct MP4MediaRecord
{
        void*   data; /* ptr to 'mdia' atom of this track */
        u32     specificInfo_size; /* size of codec specific info, in bytes */
        u32     track_count;    /* index of current track to read, 0-based. Default value is 0.
And newMedia()does not set the actual track count. */
        u32     peer_track_count; /* the peer track index, 0-based. Peer track is the track
other than current track */
        u32     frame_count;
        u32     media_width; /* video picture width in pixels */
        u32     media_height; /* video picture height in pixels */
        u32     media_duration;
        float   media_framerate;  /* average frame rate */
        u32     media_channels; /* audio channel number */
```

```
        u32     level;
        u32     profile;
        u32     track_endflag; /*whether this track is end. But the longer track's flag is
never set. So it's only indicative but not used. */
        u32     track_interleaved;
   u8      nal_length_size; /* For H264 only. Size of NAL unit length field, in bytes. */
};
typedef struct MP4MediaRecord MP4MediaRecord;
typedef MP4MediaRecord*    MP4Media;
```

3.5. eCodecType

```
typedef enum
{
        VIDEO_MPEG4  =0x20, /*Visual 14496-2 */
        VIDEO_H264   =0x21,
        AUDIO_AAC    =0x40, /* Audio 14496-3 (MPEG-4 AAC).
                        Also indicating general AAC audio type, including both MPEG-2
& MPEG-4 AAC */
        AUDIO_MPEG2_AAC = 0x66, /* Audio 13818-7 Main Profile (MPEG-2 AAC)
*/
        AUDIO_MPEG2_AAC_LC  = 0x67, /* Audio 13818-7 lowComplexity Profile
(MPEG-2 AAC LC)*/
        AUDIO_MPEG2_AAC_SSR = 0x68, /* Audio 13818-7 Scalable Sampling Rate
Profile (MPEG-2 AAC SSR)*/
        AUDIO_MPEG2  =0x69, /* Audio 13818-3 (MPEG-2 Audio)*/
   AUDIO_MP3    =0x6b, /* Audio 11172-3 (MPEG-1 Layer3)*/
        VIDEO_H263   =0xF2,
        AUDIO_AMR    =0xE1
} eCodecType;
```

4. Callback functions

4.1. Memory operation

4.1.1. callloc

```
void *MyMP4LocalCalloc(u32 TotalNumber, u32 TotalSize)
```

DESCRIPTION:
Implements the calloc function.

ARGUMENTS:
TotalNumber - total number of memory blocks
TotalSize - size of each block , in bytes

RETURN VALUE: memory pointer

4.1.2. malloc

void* MyMP4LocalMalloc (u32 TotalSize)

DESCRIPTION:
Implements the malloc function.

ARGUMENTS:
TotalSize - size of the block in bytes

RETURN VALUE: memory pointer

4.1.3. realloc

void* MyMP4LocalReAlloc (void *MemoryBlock, u32 TotalSize)

DESCRIPTION:
Implements the realloc function.

ARGUMENTS:
MemoryBlock - original memory pointer
TotalSize - size of the block , in bytes

RETURN VALUE: new memory pointer

4.1.4. free

void MyMP4LocalFree (void *MemoryBlock)

DESCRIPTION:
Implements the free function.

ARGUMENTS:
MemoryBlock - memory pointer

4.2. File Operation

4.2.1. Open file

FILE * MyMP4LocalFileOpen (const u8 *FileName, const u8 *ModeToOpen)

DESCRIPTION:
Open a file.

ARGUMENTS:

FileName – file name to open
ModeToOpen – open mode

RETURN VALUE: file pointer

4.2.2. Close file

u32 MyMP4LocalFileClose (FILE *StreamToClose).

DESCRIPTION:
Close the file.

ARGUMENTS:
fileHandle – file pointer

RETURN VALUE: Sucess or EOF.

4.2.3. Seek file

void MyMP4LocalSeekFile(FILE *fileHandle, int offset, int origin ,void *parser_context)

DESCRIPTION:
Seek a file.

ARGUMENTS:
fileHandle – file pointer
offset – file offset
parser context – application context.

4.2.4. Get file position

LONGLONG MyMP4LocalGetCurrentFilePos(FILE *fileHandle ,void *parser_context)

DESCRIPTION:
Get current file position.

ARGUMENTS:
fileHandle – file pointer
parser context – application context.

RETURN VALUE: file position.

4.2.5. Get file size

LONGLONG MyMP4LocalFileSize(FILE *fileHandle,void *parser_context)

DESCRIPTION:

Get the file size, in bytes.

ARGUMENTS:
fileHandle – file pointer
parser context – application context.

RETURN VALUE: file size in bytes

4.2.6. Read file

```
u32 MyMP4LocalReadFile (void * Buffer,
                u32       Size,
                u32       Count,
                FILE      *Stream,
                void      *parser_context)
```

DESCRIPTION:
Read data from a file

ARGUMENTS:
SourceBuffer – Pointer to a block of memory with a minimum size of (Size *Count) bytes.
Size – Size in bytes of each element to be read.
Count - Number of elements, each one with a size of “Size” bytes.
Stream – file pointer

RETURN VALUE: total bytes read successfully.

4.2.7. Write file

```
u32 MyMP4LocalWriteInFile (const void * Buffer,
                u32       Size,
                u32       Count,
                FILE      *Stream)
```

DESCRIPTION:
Write data to a file

ARGUMENTS:
Buffer – pointer to the array of elements to be written
Size – Size in bytes of each element to be written.
Count - Number of elements, each one with a size of “Size” bytes.
Stream – file pointer

RETURN VALUE: total bytes written successfully.