NO.:	VERSION:	PAGE: 1

MP4 LOW LEVEL LIBRARY PORTING GUIDE

Revision History

Revision	Date	Comments
0.1	2004/07/29	First draft

1. GENERAL DESCRIPTION

W99702 MPEG4 engine can encode and decode H.263 short header mode, MPEG4 simple profile bit-stream and includes following specification :

- Fully compliant with ISO 14492-2 : Simple Profile L0, 1, 2 & 3
- Resolutions up to 720*576 pixels
- Encodes/Decodes bit-streams up to 12Mbps
- Advanced Simple Profile Levels 0, 1, 2, 3, 4, 5
- Baseline ITU-T H.263 standard

2. SUPPORT FUNCTION

<u>Global</u>	
mp4InstallCallBackFunction	Install User's call back function
<u>MPEG4 Encoder</u>	
mp4SetEncoder	Initialize MPEG4 encoder
mp4StartEncoder	Trigger MPEG4 encoder to encode a frame
mp4ResetEncoder	Reset MPEG4 encoder
mp4SetEncQuantmatrix	Set User defined Quant table
mp4GetCurrentBitStreamAddr	Get current encoded bit-stream address
mp4GetEncBitStreamLength	Get current encoded bit-stream size
mp4SetNewYUVEncoderAddr	Set YUV raw data address
mp4SetNewEncBitStreamAddr	Set encoded buffer address
<u>MPEG4 Decoder</u>	
mp4SetDecoder	Initialize MPEG4 decoder
mp4ResetDncoder	Reset MPEG4 decoder
mp4StartEncoder	Trigger MPEG4 decoder to decode a frame
mp4GetDecBitstreamLength	Get decoded bit-stream size
mp4SetNewDecBitStreamAddr	Set decoded bit-stream address
mp4GetCurrentDisplayBufAddr	Get decoded YUV raw data address
mp4GetCurrentVTI	Get time increment of decoded frame
mp4SetDecBufferAddr	Set reference and output frame address

3. GLOBAL FUNCTION

MPEG4 encoder/decoder low level library interrupt will do the necessary action like clear interrupt status, swap buffer and so on. If a high level programmer wants to does their proprietary function, they can install the respective callback function for corresponding interrupt. For example, a high level programmer can install their callback function for encode done interrupt to save the encoded MPEG4 bit-stream to file in the callback function before next encoding trigger. Please refer following section for the detail description of interrupt.

3.1. mp4InstallCallBackFunction

This function installs callback function for MPEG4 encoder/decoder interrupt.

PVOID mp4InstallCallBackFunction(INT32 interruptSource, PVOID callbackfunction)

Parameters :

InterruptSource : Installed interrupt source

The interrupt source is defined as following for each interrupt.

```
#define        MP4_ENOCDE_COMPLETE_CALL_BACK     0
#define        MP4_ENOCDE_ERROR_CALL_BACK        1
#define        MP4_DECODE_COMPLETE_CALL_BACK     2
#define        MP4_DECODE_WAIT_CALL_BACK         3
#define        MP4_DECODE_ERROR_CALL_BACK        4
```

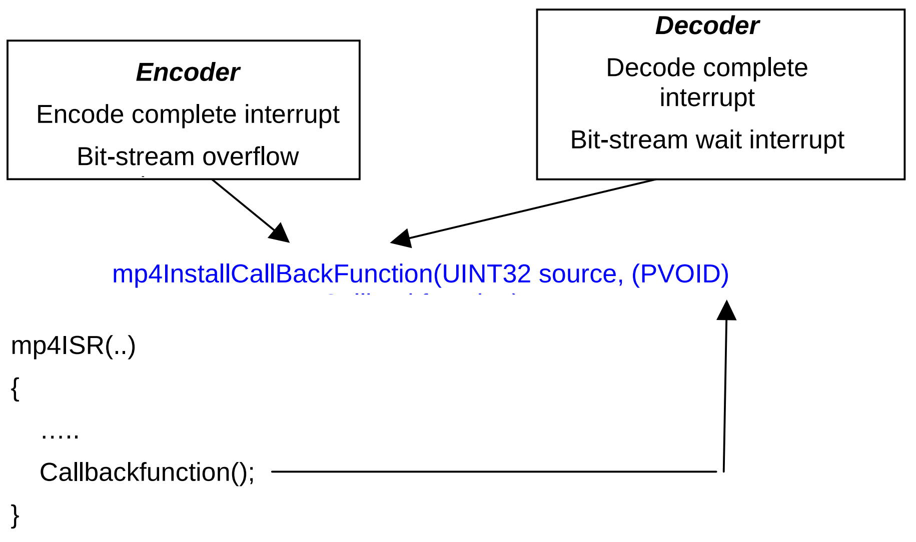
Callbackfunction : Installed callback function

Return Values : Original callback function.

Remarks : One mp4InstallCallBackFunction only installs a callback for a interrupt source. If a high level programmer wants to install multiple call back function, call this function for each interrupt source.

example :

```
    // Install userEncDone callback function for MPEG4 encode complete interrupt.
mp4InstallCallBackFunction(MP4_ENCODE_COMPLETE_CALL_BACK,(PVOID)usrEncDone);
```



Encoder
Encode complete interrupt
Bit-stream overflow
Decoder
Decode complete interrupt
Bit-stream wait interrupt
mp4InstallCallBackFunction(UINT32 source, (PVOID)

```
mp4ISR(..)
{
    …..
    Callbackfunction();
}
```

Figure 1 : Callback function in

4. MP4 ENCODER

W99702 MPEG4 engine can support to encode H.263 and MPEG4 video bit-stream with a limitation that width and height is constrained as multiple of 16.

Regarding the MPEG4 bit-stream, H/W engine only produce the VOP bit-stream. The bit-stream in other layer is produced by the W99702 firmware.

4.1. Encoder Interrupt

W99702 MPEG4 encoder may produce 2 interrupt while encoding the bit-stream. One is encode done interrupt and the other is bit-stream overflow interrupt.

Encode done interrupt occurs when the video bit-stream is produced without any error. You can get the video bit-stream from the dram address where is specified by you.

Bit-stream overflow interrupt occurs when the bit-stream size is over the region where is specified by you. If such an interrupt occurs, the bit-stream is not available for the MPEG4 decoder, you must enlarge the reserved region and wait the next encode done interrupt for the video bit-stream.

4.2. Bit Rate Control

MPEG4 low level library provides 2 bit rate control methods : **fixed quality control** and **fixed bit rate control**. They are distinguished by the field “bitrate” in _mp4Encinfo structure which is defined in mp4lib.h file and pass to mp4SetEncoder function as parameter. If bitrate is equal to 0, fixed quality control is used. Otherwise, fixed bit rate control will be adopted.

If fixed quality control is adopted, MPEG4 low level library will not control the Q value for each frame. It only use the qintra value in _mp4Encinfo structure for I frame and qinter value in _mp4Encinfo structure for P frame.

On the other hand, if fixed bit rate control is adopted, MPEG4 low level library can produce the accurate video bit-stream rate control based on the specified target bit-rate. What A high level programmer need to do is to set up the necessary information for mp4SetEncoder( ) function to initialize MPEG4 decoder before encoding. MPEG4 low level library will base on these information to do the necessary control when a VOP bit-stream is encoding.

4.3. Encoder Memory requirement

MPEG4 encoder needs following buffer for encoding as figure 2.

Input YUV buffer : contains YUV raw data for MPEG4 encoder to encode

Reference frame buffer : contains previous reconstructed YUV data for encoding P frame

Reconstructed frame buffer : contains current reconstructed YUV data

Bit-stream buffer : contains video bit-stream to decode

If all the encoded frames are I frames, the reference frame buffer is unnecessary. But this will increase total bit-stream size of a video sequence. Recommend you to provide such buffer to reduce total bandwidth.

If the encoded frame size is width * height, the buffer size for input yuv buffer, reference frame buffer and reconstructed frame buffer are width*height*1.5.

For example, if the encode frame size is 352 * 288, the each buffer size is 352*288*1.5 bytes.

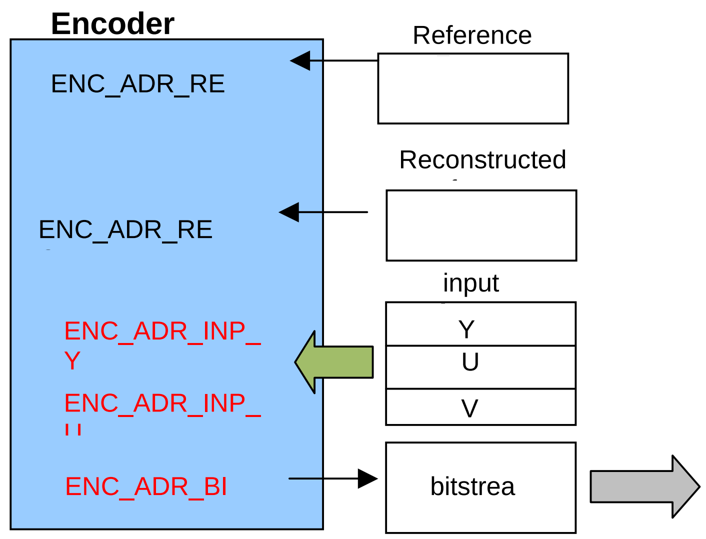


Figure 2 : Memory allocate for MPEG4

4.4. Initialize MPEG4 encoder

To initialize the MPEG4 encoder, high level programmer must pass a structure mp4Encinfo which is defined in mp4lib.h to mp4SetEncoder function to initialize the MPEG4 encoder.

The bitstreamheaderaddr field is used to store the bit-stream above VOP if the encoded bit-stream is MPEG4. And the encoded bit-stream length in bitstreamheaderaddr can be got from the returned value of mp4SetEncoder function.

If high level programmer want to overwrite the quant matrix, setting the quant_type field not equal to 0 will inform MPEG4 library to program the 128 matrix element in quant_mat array to the bit-stream instead of the default matrix.

4.5. MPEG4 bitstream

The data structure of MPEG4 visual data is as figure 3. The bit-stream above VOP is encoded by the MPEG4 initialization function “mp4SetEncoder(..)” which will return the total bit-stream size from VS to GOV layer. The VOP bit-stream is produced by each mp4StartEncoder( ) function to trigger for each frame.

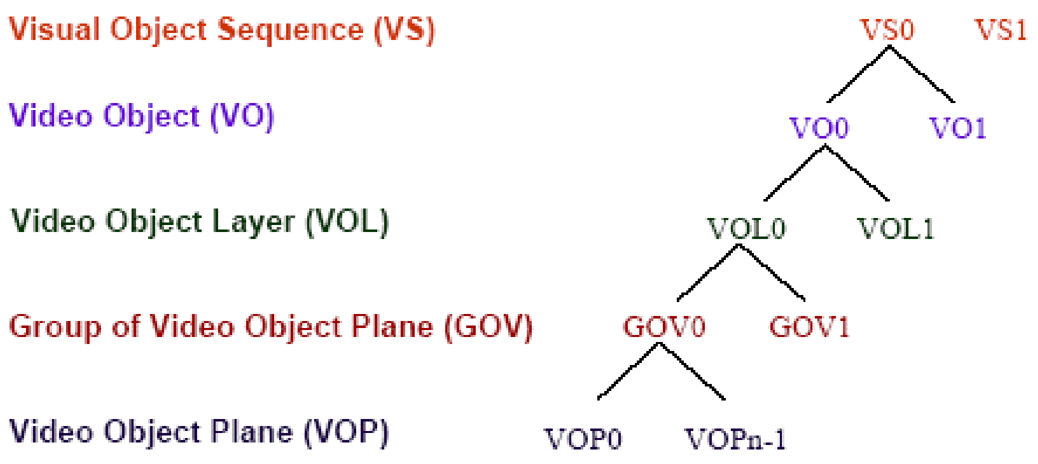


Figure 3 : Data Structure for MPEG4 Visual

4.6. MPEG4 Encoder Function

4.6.1. mp4SetEncoder

This function initializes the MPEG4 encoder.

```
UINT32 mp4SetEncoder(mp4encinfo *encinfo)
```

Parameters : pointer to a structure mp4encinfo.

Return Values : bit-stream length above VOP**.**

Remarks : The returned value is the bit-stream length in address of bitstreamheaderaddr field of mp4Encinfo structure if the encoded bit-stream type is MPEG4. The return value will be 0 if the encoded bit-stream type is H.263. In H.263 case, it means that don't care bitstreamheaderaddr field in the mp4Encinfo structure.

4.6.2. mp4ResetEncoder

This function resets MPEG4 encoder.

```
void mp4ResetEncoder(void)
```

Parameters : None

Return Values : None.

Remarks : None.

4.6.3. mp4StartEncoder

This function triggers MPEG4 encoder to encode a video bit-stream.

```
void mp4StartEncoder(void)
```

Parameters : None

Return Values : None.

Remarks : Only a frame bit-stream is encoded for each function call.

4.6.4. mp4SetEncQuantmatrix

This function sets the Quant matrix for MPEG quantisation.

```
UINT8 mp4SetEncQuantmatrix(PUINT8 matrix)
```

Parameters : pointer to 128 quant element.

Return Values : Success / Fail

Remarks : It is not commended to set the quant element by high level programmer.

4.6.5. mp4GetCurrentBitStreamAddr

This function gets the dram address of encoded bit-stream.

```
UINT32 mp4GetCurrentBitstreamAddr(void)
```

Parameters : None

Return Values : Bit-stream buffer address

Remarks : None.

4.6.6. mp4GetEncBitStreamLength

This function gets encoded bit-stream size.

```
UINT32 mp4GetEncBitstreamLength(void)
```

Parameters : None

Return Values : Encoded bit-stream length

Remarks : None.

4.6.7. mp4SetNewYUVEncoderAddr

This function sets the dram address of source YUV data.

```
UINT8 mp4SetNewYUVEncoderAddr(UINT32 yuvaddr)
```

Parameters : YUV raw data buffer address to encode

Return Values : Success / Fail

Remarks : MPEG4 encoder always use YUV420 planar raw data for the encoding source. The buffer address must be 4 bytes alignment.

4.6.8. mp4SetNewEncBitStreamAddr

This function sets new dram address to store VOP bit-stream.

```
void mp4SetNewEncBitstreamAddr(UINT32 bitstreamaddr)
```

Parameters : bit-stream buffer address

Return Values : None.

Remarks : If high level programmer use the same buffer for all encoded bit-stream, this function will need be call once only. However, high level programmer must read back the encoded bit-stream before next mp4StartEncoder function is called. Otherwise, new encoded bit-stream will overwrite the old bit-stream. The buffer address must be 4 bytes alignment.

5. MPEG4 DECODER

MPEG4 decoder can decode H.263 short header mode and MPEG4 simple profile bit-stream. Maximum resolution is up to 720*576. A high level programmer can use the decoded YUV raw data to display.

5.1. Decoder Interrupt

MPEG4 decode provides 3 decode interrupts : **decode done**, **bit-stream wait** and **bit-stream error** interrupt.

Decode done interrupt : When MPEG4 can decode the video bit-stream and don't find any error, it will issue a decode done interrupt. System can get the decoded YUV raw data after this interrupt.

Bit-stream wait interrupt : MPEG4 decoder allows system only load partial bit-stream into dram. When the partial bit-stream is decoded, MPEG4 decoder will issue a bit-stream wait interrupt to notify system to load the next partial bit-stream to decode. When all the bit-stream of a frame is decoded, it will issues a decoded done interrupt.

Bit-stream error interrupt : When MPEG4 decoder finds any error in the video bit-stream, it will issues a bit-stream error interrupt.

5.2. Decoder Memory requirement

MPEG4 decoder requires following buffers for decoding a video bit-stream as figure 4.

Reference buffer : contains YUV raw data of previous decoded frame if decoded P frame

Output buffer : contains YUV raw data of current decoded frame

Bit-stream buffer : contains video bit-stream to decode

If the decoded frame size is width * height, the buffer size for reference frame buffer and output frame buffer are width*height*1.5.

For example, if the encode frame size is 352 * 288, the each buffer size is 352*288*1.5 bytes.

These buffers address is specified while calling MPEG4 encoder.

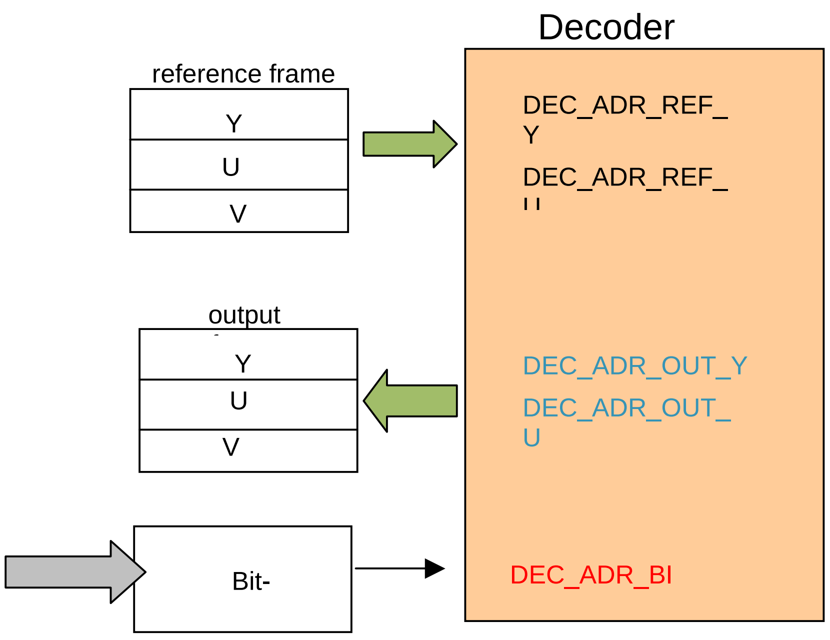


Figure 4 : Memory allocate for MPEG4

5.3. Initialize MPEG4 decoder

A high level programmer doesn't need to interpret the bit-stream before passing the bit-stream to MPEG4 decoder. Because MPEG4 low level library can decode H.263 short header mode bit-stream and MPEG4 bit-stream, you only need to load the video bit-stream to bit-stream buffer then call "mp4SetDecoder" function to initialize decoder itself. The mp4SetDecoder function will parse the bit-stream and provide necessary information to system such as pixels, lines, bit-stream codec type and so on.

A mp4decinfo structure is used to initialize the MPEG4 decoder. There are 2 fields that are specified by high level programmer :

bitstreamaddr : specify the buffer address for H.263 bit-stream or bit-stream above VOP bit-stream for MPEG4 bit-stream. If the bit-stream is MPEG4, MPEG4 decoder will parse the bit-stream until VOL is decoded. However, if the bit-stream is H.263 short header bit-stream, it only decodes the information for pixels and lines.

decodedbufsize : specify the buffers size for bit-stream.

After calling mp4SetDecoder function, high level programmer can get other useful information from other field of mp4decinfo structure.

5.4. Decode bit-stream

As Figure 5, mp4SetDecoder function will get the necessary information from the bit-stream above VOP and initialize the MPEG4 decoder itself. Mp4SetDecoder function will stop parsing the MPEG4 bit-stream when it finishes the parsing of VOL.

After calling mp4SetDecoder function, calling mp4StartDecoder for each VOP bit-stream will trigger MPEG4 decoder to decode each VOP bit-stream. If first VOP bit-stream is followed VOL, you can call mp4StartDecoder function from the VOL dram address. MPEG4 decoder will skip the VOL bit-stream and decoded followed VOP bit-stream automatically as figure 5. Otherwise, you can specify the new VOP bit-stream address by mp4SetNewDecBitStreamAddr function for each VOP bit-stream then calling the mp4StartDecoder to decode it.

If the MPEG-4 bit-stream is at the same dram address for each VOP, the mp4SetNewDecBitstreamAddr function is unnecessary. It is only necessary when the bit-stream address is different for each VOP bit-stream. When the VOP bit-stream is decoded, you can call the mp4GetDecBitstreamLength function to get the real VOP bit-stream size for decoded frame.

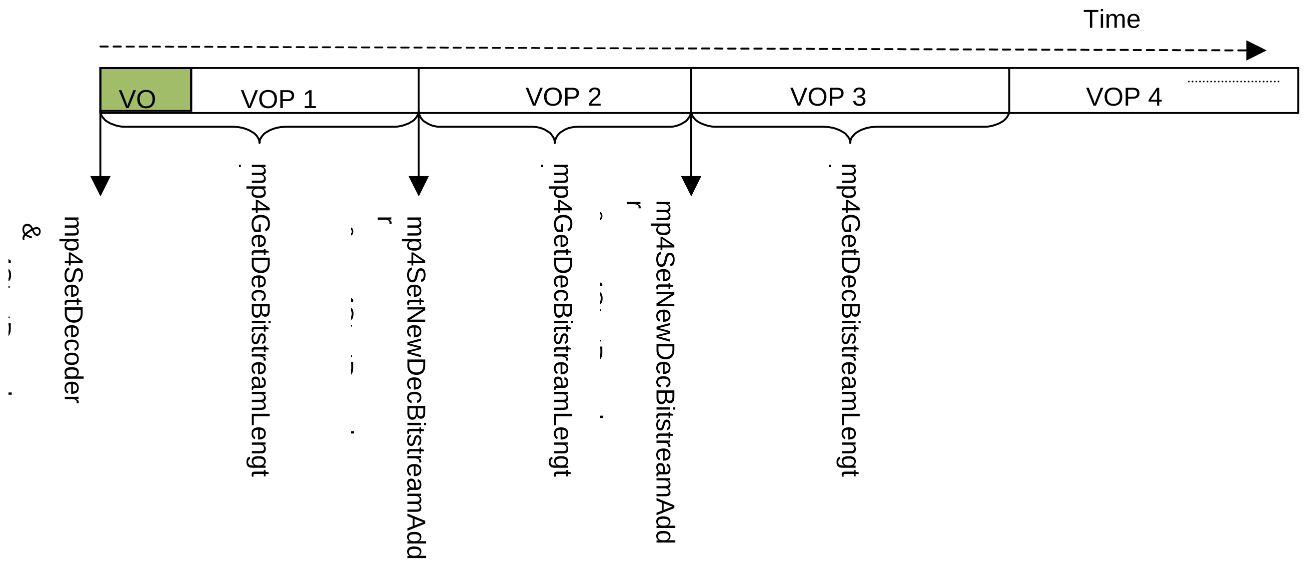


Figure 5 : Decode bit-

If your buffer size is limited, you also can partition a VOP bit-stream into several parts. As figure 6, a VOP bit-stream is partitioned into 3 parts. Call mp4StartDecoder function at the beginning of a VOP bit-stream. When the specified buffer size is consumed, a decode wait interrupt will inform you to prepare the next part of bit-stream. You can hook your callback function into the decode wait interrupt and read the next partition in the hooked callback function. When callback function is returned, MPEG4 low level library will continue the decoding until partition bit-stream consumed or completed.

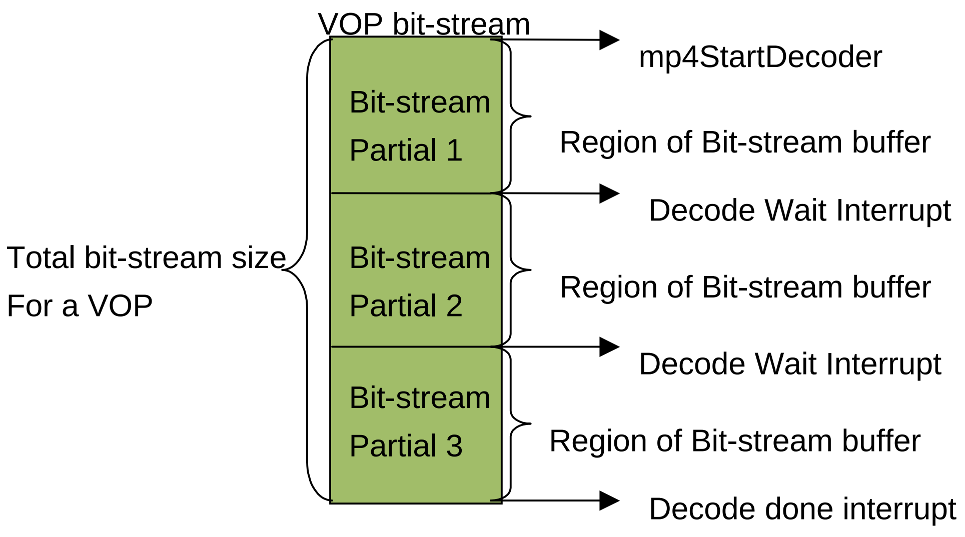


Figure 6 : Decode wait

5.5. MPEG4 decoder function

5.5.1. mp4SetDecoder

This function initializes the MPEG4 decoder.

```
void mp4SetDecoder(mp4decinfo *decinfo)
```

Parameters : pointer to a structure mp4decinfo

Return Values : None.

Remarks : mp4decinfo structure is defined in mp4lib.h file. By this function, high level programmer can get useful information for this bit-stream. For example, H.263 or MPEG4 bit-stream is decided by short_header variable. (0:H.263, 1:MPEG4 bit-stream)

5.5.2. mp4ResetDecoder

This function resets MPEG4 decoder.

```
void mp4ResetDecoder(void)
```

Parameters : None

Return Values : None.

Remarks : In normal decoding processing, it is unnecessary to reset decoder by this function. MPEG4 library will reset decoder in the ISR by itself.

5.5.3. mp4StartDecoder

This function triggers MPEG4 decoder to decode a VOP frame.

```
void mp4StartDecoder(void)
```

Parameters : None

Return Values : None.

Remarks : Only a VOP bit-stream is decoded for this function. For successive VOP bit-stream, a sequence of mp4StartDecoder are called to decoded these VOP bit-stream.

5.5.4. mp4GetDecBitstreamLength

This function gets bit-stream size of current decoded VOP.

```
UINT32 mp4GetDecBitstreamLength(void)
```

Parameters : None

Return Values : decoded mpeg4 bit-stream length.

Remarks : Returned length include VOP bit-steam and extra bytes if decoded bit-stream buffer contains extra byte before VOP bit-stream.

5.5.5. mp4SetNewDecBitStreamAddr

This function sets buffer address for VOP bit-stream.

```
UINT8 mp4SetNewDecBitStreamAddr(UINT32 streamaddr)
```

Parameters : buffer address of bit-stream

Return Values : Success / Fail.

Remarks : If the high level programmer always read the bit-stream on the same buffer address, it is unnecessary to call this function to set the buffer address again. The buffer address must be 4 bytes alignment.

5.5.6. mp4GetCurrentDisplayBufAddr

This function gets the decoded YUV buffer address.

```
UINT32 mp4GetCurrentDisplyBufAddr(void)
```

Parameters : None.

Return Values : Buffer address of decoded YUV raw data.

Remarks : The reference and output address will be swapped by MPEG4 library. It is necessary to call this function to get the decoded YUV buffer address for the current bit-stream.

5.5.7. mp4GetCurrentVTI

This function gets the time increment of current decoded VOP.

```
UINT32 mp4GetCurrentVTI(void)
```

Parameters : None.

Return Values : Time increment**.**

Remarks : High level programmer can decide the display frame rate by following equation :

fps = vop_time_increment_resolution / difference of two bit-stream's vop_time_increment

where vop_time_increment_resolution is a field got from mp4SetDecoder function and vop_time_increment is got from this function.

5.5.8. mp4SetDecBufferAddr

This function gets the time increment of current decoded VOP.

UINT8 mp4SetDecBufferAddr(UINT32 ref_addr, UINT32 out_addr)

Parameters :

ref_addr : reference frame buffer address

out_addr : output frame buffer address

Return Values : Success/ Fail

Remarks : This function must be called after mp4SetDecoder function. The out_addr address is not always used for output buffer. MPEG4 library may swap the ref_addr and out_addr for the output buffer on purpose. To retrieve the real output buffer address, the mp4GetCurrentDisplaybufAddr function is used. These buffer addresses must be 4 bytes alignment.

6. SAMPLE CODE

6.1. Sample code to install MPEG4 callback function

```
int main(void)
{
//------------------- initalize Interrupt -----------------------------------
//Set the interrupt is high/low level trigger or positive/negative edge trigger
WB_InstallISR(IRQ_LEVEL_1, IRQ_MPEG, (PVOID)MP4_Handler);       //
WB_SetGlobalInterrupt(DISABLE_ALL_INTERRUPTS);
WB_SetAIC2SWMode();                                    // Set AIC to SW Mode
//------------------- initalize System setting for MPEG 4 Engine-------------------------
initSystem4MP4();
//------------------- Test MPEG 4 item-------------------------
systemiInitMP4Encoder();
systemInitMP4Decoder();

mp4InstallCallBackFunction(MP4_ENCODE_COMPLETE_CALL_BACK,(PVOID)EncDoneCallback);
mp4InstallCallBackFunction(MP4_ENCODE_ERROR_CALL_BACK,(PVOID)EncErrorCallback);
mp4InstallCallBackFunction(MP4_DECODE_COMPLETE_CALL_BACK,(PVOID)DecDoneCallback);
mp4InstallCallBackFunction(MP4_DECODE_WAIT_CALL_BACK,(PVOID)DecWaitCallback);
mp4InstallCallBackFunction(MP4_DECODE_ERROR_CALL_BACK,(PVOID)DecErrorCallback);

WB_EnableInterrupt(IRQ_MPEG);                          // Enable MP4 Interrupt
WB_SetLocalInterrupt(ENABLE_IRQ);              // enable CPSR I bit

mp4StartEncoder();                  // If need to encode a MPEG4 bit-stream now

while(1);
}
```

6.2. Sample code to initialize MPEG4 encoder

```
void systemiInitMP4Encoder (void)
{

        mp4Encinfo mp4encinf;
        UINT32 headbitstreamlength;

        memset(&mp4encinf,0,sizeof(mp4encinf));

        mp4encinf.vop_time_increment_resolution = 30000;
        mp4encinf.video_packet_interval = 0;          //0:disable ,others specify Video Packet Interval
        mp4encinf.bitrate = 768*1024;
        mp4encinf.baserate = 30;
        mp4encinf.qintra = mp4encinf.qinter = 6;
        mp4encinf.pixels =176;
        mp4encinf.lines = 144;
        mp4encinf.quant_type = 0;                          // 0: H.263Q, 1: MPEG4 Q
        mp4encinf.PframenumbeteweenIfram = 100;            // -1 : I frame then all P frame
        mp4encinf.referenceaddr = 0x400000;                  //4M addr
        mp4encinf.reconstructaddr = 0x500000;                //5M addr
        mp4encinf.bitstreamheaderaddr = 0x600000;            //6M addr
        mp4encinf.codectype = MPEG4_BITSTREAM;               // MPEG4 : 1, H.263 : 0

        headbitstreamlength = mp4SetEncoder(&mp4encinf);
        mp4SetNewYUVEncoderAddr(0x100000);                   //1M addr
        mp4SetNewEncBitStreamAddr(0x600014);                 // real VOP bit-stream
        // mp4StartEncoder();    // if need to start MPEG4 encoder here

}
```

6.3. Sample code to initialize MPEG4 decoder

```
void systemInitMP4Decoder (void)
{
        mp4decinfo decinfo;
        memset(&decinfo,0,sizeof(decinfo));

        decinfo.bitstreamaddr   = 0x600000;      // 6M address
        decinfo.decodedbufsize  = 0x80000;       // 512K buffer size

        mp4SetDecoder(&decinfo);
        mp4SetDecBufferAddr(0x400000,0x700000);     // 4M & 7M

        mp4SetNewDecBitStreamAddr(0x680000);        // 6.8M
        // mp4StartDecoder();    // if need to start MPEG4 decoder here

}
```