VisualOn WMA Decoder Reference Manual

VisualOn, Inc.

2959 S. Winchester Boulevard, Suite 201A
Campbell, CA 95008, USA
http://www.visualon.com

Revision History

Date	Version	Changes	Author
Sep 21 2009	1.0.0	Initial Version	Witten Wen

Table of Contents

1 Overview

This documentation details the Application Programming Interfaces (APIs) of VisualOn WMA (Windows Media Audio) decoder. VisualOn WMA SDK supports WMA Standard, WMA Professional, WMA Professional Plus (LBR v1, v2 and v3) and WMA lossless. It allows you to decompress standard WMA raw data bit-streams to PCM format data. And support down multichannel to stereo, and resample sample rate. We support Windows XP, PPC2003, Windows Mobile, Linux, Android os and have optimized version on armv4, armv6, and neon platform. A license file is used to activate the support of different combination of decoders in the release.

The following figure 1 summarizes the valid sequences of execution of the functions for a audio decoder instance.

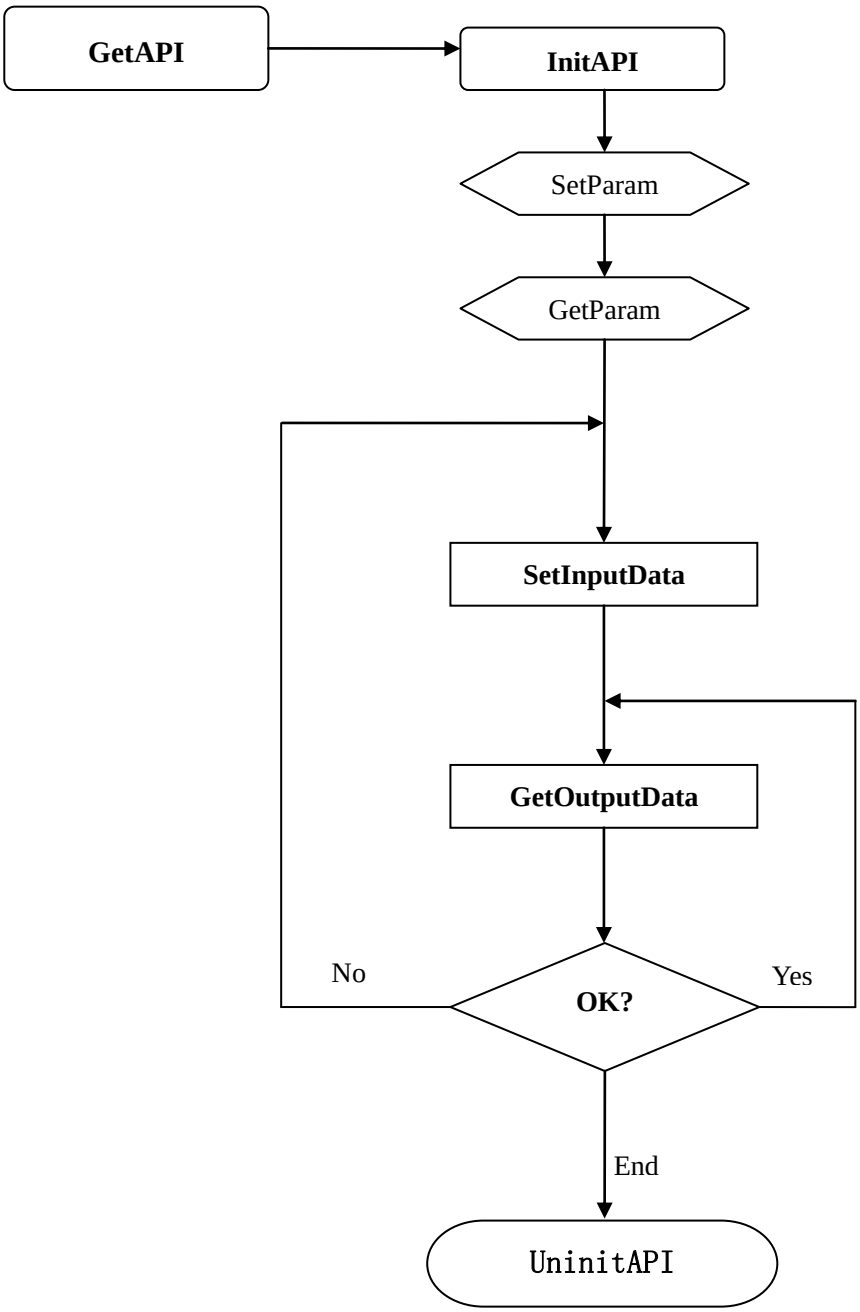

Figure 1. Audio decoder figure.

2 Files In SDK

2.1 Header files

1) Common header files used by other SDK: voType.h, voAudio.h
2) Specified header file used by MPEG4 decoder: voWMA.h

2.2 Sample file

Sample application: voWMA_D_SAMPLE.c

2.3 Lib files

Lib files for core decoder: voWMADec.*

It may include other files for convenient debugging.

3 Decoder control API

3.1 Common structure

3.1.1 STRUCTURE VO_CODECBUFFER

It is used for input or output data buffer setting, the declaration is following:

```
typedef struct {
    VO_PBYTE  Buffer;          /*!< Buffer pointer */
    VO_U32    Length;          /*!< Buffer size in byte */
    VO_S64    Time;            /*!< The time of the buffer */
} VO_CODECBUFFER;
```

3.1.2 STRUCTURE VO_AUDIO_OUTPUTINFO

It is used for get audio data information, include channel, samplerate, and buffer used, and a reserve value included. the declaration is following
typedef struct

```
{
    VO_AUDIO_FORMAT     Format;         /*!< Sample rate */
    VO_U32              InputUsed;      /*!< Channel count */
    VO_U32              Resever;        /*!< Resevered */
} VO_AUDIO_OUTPUTINFO;
```

The structure VO_AUDIO_FORMAT is for audio information, include channel, samplerate, and sample bits. the declaration is following

```
typedef struct
{
    VO_S32 SampleRate;   /*!< Sample rate */
    VO_S32 Channels;     /*!< Channel count */
    VO_S32 SampleBits;   /*!< Bits per sample */
} VO_AUDIO_FORMAT;
```

3.1.3 STRUCTURE VO_CODEC_INIT_USERDATA

It is used for init decoder to set some parameter, like memory operator. the declaration is following:

```
typedef struct{
    VO_INIT_MEM_FlAG        memflag;    /*!<memory flag    */
    VO_PTR                  memData;    /*!<a pointer to VO_MEM_OPERATOR
or a preallocated buffer    */
    VO_U32                  reserved1;  /*!<reserved    */
    VO_U32                  reserved2;  /*!<reserved */
}VO_CODEC_INIT_USERDATA;
```

The enum VO_INIT_MEM_FlAG is discript init parameter type, Now there are only two type, the audio data just use the first type VO_IMF_USERMEMOPERATOR. the declaration is following:

```
typedef enum{
    VO_IMF_USERMEMOPERATOR      =0,  /*!< memData is  the pointer of
                                        memoperator function*/
    VO_IMF_PREALLOCATEDBUFFER   =1,  /*!< memData is  preallocated
                                        memory*/
    VO_IMF_MAX = VO_MAX_ENUM_VALUE
}VO_INIT_MEM_FlAG;
```

3.1.4 ENUM VO_AUDIO_CODINGTYPE

It is audio type declaration what we have support, when init some decoder, you should set it into the decoder for mutli instance debug. the declaration is following:

```
/**
  *Enumeration used to define the possible audio codings.
  */
typedef enum VO_AUDIO_CODINGTYPE {
      VO_AUDIO_CodingUnused = 0, /**< Placeholder value when coding is N/A    */
      VO_AUDIO_CodingPCM,          /**< Any variant of PCM coding */
      VO_AUDIO_CodingADPCM,      /**< Any variant of ADPCM encoded data */
      VO_AUDIO_CodingAMRNB,      /**< Any variant of AMR encoded data */
      VO_AUDIO_CodingAMRWB,      /**< Any variant of AMR encoded data */
      VO_AUDIO_CodingAMRWBP,    /**< Any variant of AMR encoded data */
      VO_AUDIO_CodingQCELP13,     /**< Any variant of QCELP 13kbps encoded data */
      VO_AUDIO_CodingEVRC,         /**< Any variant of EVRC encoded data */
      VO_AUDIO_CodingAAC,          /**< Any variant of AAC encoded data, 0xA106 -
ISO/MPEG-4 AAC, 0xFF - AAC */
      VO_AUDIO_CodingAC3,           /**< Any variant of AC3 encoded data */
      VO_AUDIO_CodingFLAC,          /**< Any variant of FLAC encoded data */
      VO_AUDIO_CodingMP1,           /**< Any variant of MP1 encoded data */
      VO_AUDIO_CodingMP3,           /**< Any variant of MP3 encoded data */
      VO_AUDIO_CodingOGG,           /**< Any variant of OGG encoded data */
      VO_AUDIO_CodingWMA,           /**< Any variant of WMA encoded data */
      VO_AUDIO_CodingRA,             /**< Any variant of RA encoded data */
      VO_AUDIO_CodingMIDI,           /**< Any variant of MIDI encoded data */
      VO_AUDIO_CodingDRA,           /**< Any variant of dra encoded data */
      VO_AUDIO_Coding_MAX       = VO_MAX_ENUM_VALUE
} VO_AUDIO_CODINGTYPE;
```

3.2 Input and Output type

3.2.1 Input:

VisualOn WMA decoder support WMA Standard, WMA Professional, WMA Professional Plus, and WMA lossless. Table 1 is the details of the decoder supporting format.

Table 1: The format which VisualOn decoder supporting.

format	channel	Sampling rate (kHz)	bit
Standard	1-2	8-48	16
Pro	2-8	32-96	16, 24
Pro plus	2-8	32-96	16, 24
Lossless	2, 6	44.1-96	16, 24

3.2.2 Output

You could get the PCM data and PCM length, the PCM data is always interleaved.
And get the WMA data channel information, sample rate , and buffer used.

And you can call SetParam to set the Parameter ID to decoder so as to down channel, resample sample rate and down bit from 24 to 16. And if don't set the following parameter, the decoder will output default data format.

PARAM ID	DEFAULT	DESCRIPTION
VO_PID_WMA_SUPPTHISMPLRT	Resample more than 48 khz to half of original	Output original sample rate
VO_PID_WMA_SUPPORT24BIT	down 24 bit to 16 bit	Output original bit
VO_PID_WMA_SUPPTMTCHANL	Down multichannel to 2 channel	Output original channel

3.3 Parameter ID

VO_PID_COMMON_HEADDATA

Setting the head data VO_WAVEFORMATEX in track to the decoder, The parameter is the struture of VO_CODECBUFFER, It should be set before the decoding when decoding the wma raw data.

VO_PID_COMMON_FLUSH

Reset the decoder when seeking or restart, the parameter is NULL.

VO_PID_AUDIO_FORMAT
Getting audio format, the parameter is the struture of VO_AUDIO_FORMAT.

VO_PID_WMA_SUPPTHISMPLRT
Setting to support high sample rate, the parameter is NULL. If not set this parameter, the decoder will resample high sample rate to less than 48khz.

VO_PID_WMA_SUPPORT24BIT
Setting to support 24 bits, the parameter is NULL. If not set this parameter, the decoder will reconstruct 24 bit PCM to 16 bit PCM.

VO_PID_WMA_SUPPTMTCHANL
Setting to support multi-channel, the parameter is NULL. If not set this parameter, the

decoder will mix multi-channel to stereo.

VO_PID_WMA_OUTBUFFERSIZE
Getting WMA out buffer size, the parameter is integer.

VO_PID_WMA_FRAMELENGTH
Getting WMA max frame length, the parameter is integer.

3.4 Return code

There are some return code, The description is follow.
Table 4: return code

Return Code ID	Description
VO_ERR_NONE	Process data successful
VO_ERR_FAILED	Process data failed
VO_ERR_OUTOF_MEMORY	The memory is not enough
VO_ERR_NOT_IMPLEMENT	No support some feature
VO_ERR_INVALID_ARG	Error input parameter
VO_ERR_INPUT_BUFFER_SMALL	Input buffer small, you should input new data
VO_ERR_OUTPUT_BUFFER_SMALL	Output buffer small, you should remalloc big buffer
VO_ERR_WRONG_STATUS	The decoder status is wrong
VO_ERR_WRONG_PARAM_ID	The parameter is wrong
VO_ERR_LICENSE_ERROR	License error, you should get the new license.
VO_ERR_AUDIO_UNSCHANNEL	Unsupport channel
VO_ERR_AUDIO_UNSSAMPLERATE	Unsupport samplerate
VO_ERR_AUDIO_UNSFEATURE	Unsupport some feature
VO_ERR_WMA_NOTSUPPORT	The decoder initialization parameter is not supported
VO_ERR_WMA_INSIZENOTnBLOCKALIGN	input data size need equale to nBlockAlign or n * nBlockAlign

4 How To Build A Sample Application

4.1 Support OS and Platform

1) OS: WindowsXP, WM5.0, PPC2003, Linux, Android
2) X86, ARMv4, ARMv5, ARMv6, ARMv7(NEON)

4.2 Windows

4.3 Linux

5 Understanding The APIs

5.1 Only one API

VO_S32 VO_API voGetWMADecAPI (VO_VIDEO_DECAPI * pDecHandle,

VO_U32 uFlag)

To simplify the interface, we only provide one API for this SDK. Decoder will fill handle VO_AUDIO_CODECAPI * pDecHandle. Actually, structure VO_AUDIO_CODECAPI (refer to voAudio.h) will provide six functions for detail decoding process.

5.2 Six working functions

1) VO_U32 Init (VO_HANDLE * phDec,

VO_AUDIO_CODINGTYPE vType,

VO_CODEC_INIT_USERDATA * pUserData);

Init the audio decoder module and return decorder handle.

phCodec [OUT] Return the audio codec handle

vType [IN] The codec type if the module support multi codec.

pUserData [IN] The init param. It is memory operator or alloced memory

if return VO_ERR_NONE Succeeded,else failed.

Note:

a) For every decoder instance, you have to call it first.

b) Through configure VO_CODEC_INIT_USERDATA, Internal memory used by decoder can be allocated by application.

2) VO_U32 SetInputData (VO_HANDLE hDec,

VO_CODECBUFFER * pInput);

Set compressed audio data as input.

hCodec [IN]] The Codec Handle which was created by Init function.

pInput [IN] The input buffer param.

if return VO_ERR_NONE Succeeded,else failed.

Note:

For now, this SDK only supports block input mode. You can set block data together. The SDK also supports inputting one or several block.

3) VO_U32 GetOutputData (VO_HANDLE hDec,

VO_CODECBUFFER *pOutBuffer,

VO_AUDIO_OUTPUTINFO * pOutInfo);

Get the uncompressed pcm audio data and audio information. The structure VO_AUDIO_OUTPUTINFO is from voAudio.h

hCodec [IN]] The Codec Handle which was created by Init function.

pOutBuffer [OUT] The output audio data

pOutInfo [OUT] The dec module filled audio format and used the input size.pOutInfo->InputUsed is total used the input size.

If return VO_ERR_NONE Succeeded. Else if VO_ERR_INPUT_BUFFER_SMALL The input was finished or the input data was not enought. Else failed.

Note:

This function will output one frame pcm data. You can continue to call it until decoding all the blocks data set by SetInputData().

4) VO_U32 SetParam (VO_HANDLE hDec,

VO_S32 uParamID,

VO_PTR pData);

Set the parameter for special target.

hCodec [IN]] The Codec Handle which was created by Init function.

uParamID [IN] The param ID.

pData [IN] The param value depend on the ID

if return VO_ERR_NONE Succeeded,else failed.

Note:

Application can configure decoder behavior through it.(We can customize it for You, if you have any special requirement)

5) VO_U32 GetParam (VO_HANDLE hDec,

VO_S32 uParamID,

VO_PTR pData);

Get the parameter for special target.

hCodec [IN]] The Codec Handle which was created by Init function.

uParamID [IN] The param ID.

pData [IN] The param value depend on the ID

if return VO_ERR_NONE Succeeded, else failed.

Note:

Application can get internal information of decoder through it. (We can customize it for You, if you have any special requirement)

6) VO_U32 Uninit (VO_HANDLE hDec);

Un-initialize the decoder.

hCodec [IN]] The Codec Handle which was created by Init function.

if return VO_ERR_NONE Succeeded,else failed.

6 Understanding Sample Code

6.1 Memory

1) Input memory:

   Memory used by compressed audio data is allocated by application. It can cooperate with parser.

2) Decoder Internal memory:

   There are three methods to provide the internal memory used by decoder.

   a) Default method.

      Decoder call system function malloc() to malloc memory.

   b) Application provide memory operation functions

      Application can set VO_MEM_OPERATOR(defined in voMEM.h) to decoder when initialization.

6.2 Input mode

We support only one input mode:

a) Frame mode

   When calling SetInputData(), the input data should be one or more completed block data.

6.3 Decoding process

For details, please check the comments in sample code.

7 Support

If you have any problem about this SDK, please feel free to contact info@visualon.com.