

# Digital Multimedia A Case Study

Copyright © 2008 Cewidus Technologies Private Limited. All rights reserved.

The information contained in this document represents the current view of Cewidus on the issue discussed as of the date of publication. Because Cewidus must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Cewidus, and Cewidus cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for information purposes only. Cewidus makes no warranties, express or implied, in this document.

---

## Multimedia

Multimedia is a combination of various media type contents such as audio, video, animation, graphics combined to entertain the audience. It involves processing of these various media contents. Multimedia is present in a wide variety of devices around us. Mobile phones, Portable Media Players (PMP), MP3 players, DVD players, Digital television/Set Top Box and many more provide access to multimedia content.

This document describes in brief the various terminologies used in multimedia. Typical multimedia player application architecture is also discussed in this document.

---

**Figure 1**  
Multimedia Devices



---

## Multimedia Content

The various content types that can be part of a multimedia presentation are:

### **Video/Image**

This includes still picture content and full motion video. Image standards include JPEG, GIF, PNG, TIFF etc. Motion video standards used are MPEG-1/2/4, H.263, H.264, WMV, RV, DivX etc. MPEG-1 is used for storing audio video content in VCDs. MPEG-2 is used in DVD. MPEG-4, H.264 are newer standards used in various streaming applications as they provide high quality video at low bit rates. WMV (Windows Media Video) is developed by Microsoft. Real Video is used by Real Player for streaming video content.

### **Audio**

This includes compressing audio data which includes speech, music. The popular standards are MP3, AAC, WMA. Real Networks has Real Audio format.

### **Other types**

The other content types that are combined with audio video data are Text content such movie subtitles, EPG information etc. These content are added along with audio video data to provide more information and interactivity to users.

---

## Multimedia Formats

The audio video data is stored in many different formats based on applications. The common format categories are:

### **File based (storage)**

The audio video data stored in a physical media falls under this category. The formats commonly used are MP-4, ASF, and AVI. VCDs use MPEG-1 format where as DVDs use MPEG-2 formats to store audio video content along with other contents such as sub-titles and navigation information.

### **Streaming**

The audio video contents can also be streamed over Internet using RTP. Digital Broadcast is also used to provide audio video contents to public. This can be through terrestrial broadcast, coaxial cable network or from satellite using a small dish. All these require a set top box unit to receive the content.

---

## Multimedia Framework

The audio video data in various formats can be played back on a PC or Handheld devices. These require the player application. Some popular architecture frameworks used in such applications are *Directshow*, *OpenMAX*, *xDM*.

### **DirectShow**

Directshow is Microsoft architecture for multimedia applications. Microsoft Windows provides set of APIs and COM based framework for capture as well as playback of multimedia content. It supports a wide variety of formats such as ASF, MPEG, AVI, MP3 and WAV sound files. DirectShow is based on Component Object Model (COM). The architecture specifies an entity called Filter, which represents one stage of data processing in the multimedia application.

Example of a Filter is the Decoder filter which decodes the data. The various filters are connected in sequence to realize the multimedia application. The Filter has input and/or output *pins* which is used to connect it to other filters. The connection across the pin is based on the media type of the pin. The connection of various filters forms a Filter Graph. The main types of Filters are:

1. Source filters: This filter provides the source stream of data. E.g. reading raw data from a file. This filter has only output pin(s).
2. Transform filter: This filter transforms the data that is given as input and provides it to output pin. Example is a decoder. It will have input and output pins.
3. Renderer filter: This filter renders the data. Example is audio renderer which sends the data to the sound card.

Directshow in Windows provides a set of default filters such as Async source filter, transform filter for decoders such as mp3, wmv, wma, parser filters for formats such as ASF, AVI and renderer filters for audio and video as well. To develop a multimedia application using DirectShow, one can use pre existing filters, or write custom filters for the required media formats. Then build the graph using these filters and invoke the media control APIs to run the application.

### **OpenMAX**

OpenMAX is a royalty free, cross platform interfaces for multimedia applications. OpenMAX is developed by Khronos Group. OpenMAX provides three layers of interfaces for developing multimedia applications, namely Application layer (AL), Integration Layer (IL), Development Layer (DL).

OpenMAX AL provides a standardized interface between an application and the multimedia middleware. It helps in porting the applications to other platforms which are OpenMAX AL compliant.

OpenMAX IL provides a standardized interface for audio, video and image codecs. It helps in combining media frameworks and applications with various multimedia codecs in a unified manner.

OpenMAX DL acts as an interface between the physical hardware such as DSP, hardware accelerators and the multimedia codecs. This helps in porting and optimizing the codecs to different hardware platforms.

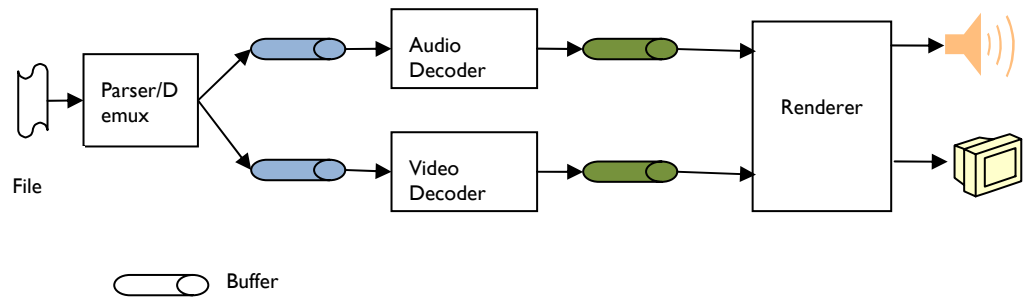
### **xDM**

Texas Instruments eXpressDSP Digital Media framework (xDM) and eXpressDSP Algorithm Interoperability Standard (xDAIS) provide a set of coding conventions and APIs that enable algorithms to be integrated quickly. By establishing standards for algorithms used on TI DSPs, both algorithm creator and system integrator are benefited. xDM APIs are defined for various codec classes such as Video decoder/encoder, Audio decoder/encoders. xDM enables application framework to support multiple codec formats of the same type without many modifications.

## Multimedia Player Application

Lets us consider a typical multimedia application, a Player. The components required to form a Player application are:

1. Demux/parser: These components are capable of reading media formats such as AVI which contain audio and video data, and demultiplex the audio and video streams.
2. Decoder: The decoder is capable of uncompressing the particular audio or video data.
3. Renderers: The renderer is capable of passing on the uncompressed audio video data to the hardware for final rendering.



**Figure 2**

A typical block diagram of a Player

## Functioning of the Player

The AVI parser can read the input file and demux the audio video data streams. It places the audio data in the audio decoder buffer and video data in the video decoder buffer. This data is usually placed in the buffer in packets containing one frame of data. However, certain file formats have one frame data broken into chunks, in which case the decoder has to read multiple chunks of data to get one frame. The parser also places the time stamps associated with each audio/video frame along with the frame data.

The audio decoder reads the frame data from its input buffer and generates the decompressed PCM data, which is placed in the decoder output buffer. The video decoder decodes the video data from its input buffer and places the uncompressed video frame data onto its output buffer. Video decoder output format could be RGB data or YUV data. The decoded audio data is continuous stream of PCM data without time stamps. Whereas the video data has time stamp associated with it, which indicates the time at which it has to be rendered. This is used in Audio-Video synchronization, which will be discussed in the following sections.

The Renderer module is responsible of reading the uncompressed data from the decoder output buffers and passing it onto Audio device and Video device. It also has to synchronize the audio and video data so that the viewer is able to view the content with proper Lip sync. The renderer module could be split into separate renderers for audio and video streams. In more complex systems, the renderer is capable of identifying skew between audio and video streams as well as skew between the rate of consumption of data by the hardware and the rate of generation of data by the upstream modules. This information can be transmitted upstream, so that the other modules can take necessary steps to control the data rate to make the presentation less annoying.

---

## Buffering

The buffers between demux – decoder and decoder – renderer not only help in splitting the audio and video data streams, but help in controlling the rate of data flowing through the systems. By their nature, the audio and video have different data rates. The decoders (esp. video decoder) require different time for different frames depending on the frame complexity. The hardware devices, however, consume data at fixed rate. The audio DAC consumes the data at the sampling rate, where as the video controller refresh the screen at a fixed frame rate. Buffers help in smoothing these data rate differences. In case multiple threads are used in the application, buffers are required to transfer data across the threads. The Buffers could be simple cyclic buffer, or more complex buffers in case of multi-threaded application.

---

## Audio Video Synchronization

The Audio Video synchronization or Lip sync refers to the timing relation between audio and video data streams of a multimedia presentation. The timing as captured by the encoder has to be maintained during playback; otherwise the audio and video will be out of sync and may be annoying to the viewer. The encoder or multiplexer adds time stamps to the encoded data frames, which is available in the file. The Parser or Demux gets the time stamps and passes it through the decoder to the renderer which uses it while rendering. In case of Video, the renderer matches the time stamp (PTS) of each decoded frame with the system clock and renders the frame at the appropriate time. Usually in the case of audio data, being a continuous stream of PCM samples, time stamp is not used while rendering. Though, the audio PTS can be used to detect skew in the system.

### Skew

Skew can occur when the system clock and audio clock are not generated from a common source. In this case, the audio clock driving the DAC, may slowly drift away from the system clock, which is used by the renderer. This will cause gradual loss of AV sync. This skew problem can be corrected by the following methods.

### **Audio Master**

Since a distortion in audio is easily noticed by the viewer, usually audio stream is given more importance. This is referred as Audio Master. In this system, the audio clock (DAC clock) drives the playback. Meaning, the audio clock is used as system clock. The decoder Audio PCM data is continuously pushed to the DAC, while the Video frames are then synced to the audio data. If a video frame arrives late due to more decoding time, it will be dropped, thus maintaining AV sync.

### **Video Master**

In this system, the Video data is considered as master. The video frames are rendered as per the system clock. The audio data is rendered by comparing its time stamps with the system clock. If there is a break in video due to network delay or decoding delay, the system clock is adjusted so that the video frames are not dropped. The audio will be muted or silenced when there is a break in video till audio gets in sync with video.