

## Geode™ SC1200/SC1210 IAOC Devices: Digital Versatile/Video Disk (DVD) API Implementation

### 1.0 Introduction

National Semiconductor® has developed a Linux Digital Video API (application programming interface) for set-top box platforms based on National's Information Appliance On a Chip (IAOC) devices: Geode™ SC1200 Set-Top Box On a Chip and SC1210 Multimedia System On a Chip.

The DVD subsystem employs the MPEG-2 (Motion Picture Expert Group) decoder, based on Sigma Designs' EM8400 MPEG-2 decoder chip.

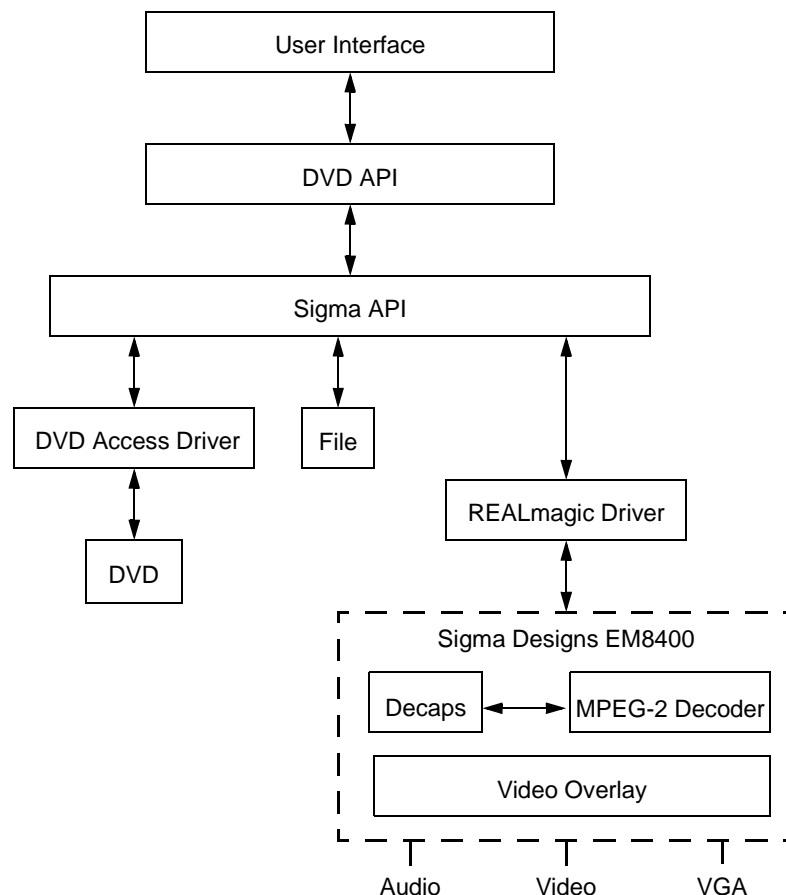
The EM8400 consists of a driver level (kernel) interface to the hardware and upper layer. The upper layer traps the interrupt raised by the driver (in the user space), and data is processed in the user space. The DVD API calls the DVD Navigation and control functions of Sigma Designs' API.

### 2.0 Discussion

This document describes the implementation of the DVD (Digital Versatile/Video Disk) subsystem in the Digital Video API.

The figure below depicts the DVD API software architecture and hardware interface in the DVD subsystem.

### Linux Digital Video Application Programming Interface - DVD Subsystem



## 3.0 DVD API Functions

### 3.1 VMW\_DVD\_ACTIVATEMENU

**Description:**

Display the DVD menus.

**Syntax:**

```
U32 VMW_DVD_ActivateMenu(DVDMenu MenuID);
```

**Parameters:**

Parameter	Description
MenuID	Menu ID to activate the following: DVD_MENU_ID_TITLE DVD_MENU_ID_ROOT DVD_MENU_ID_ANGLE DVD_MENU_ID_AUDIO DVD_MENU_ID_PART DVD_MENU_ID_SUBPICTURE DVD_MENU_ID_RESUME

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_MENU	On failure

**Implementation:**

VMW\_DVD\_Menu function is wrapped around Sigma Designs' MenuCall function. The MenuCall function displays the menu of the selected category on the disk, and will display all titles of the disk, or parts of titles (as requested), depending on the menuID flag.

**Pseudocode:**

```
U32 VMW_DVD_Menu(DVDMenu MenuID)
{
    return MenuCall(MenuID);
}
```

**Also see:**

None

### 3.2 VMW\_DVD\_ANGLESELECT

**Description:**

Select the specified stream for playback.

**Syntax:**

```
U32 VMW_DVD_AngleSelect(U32 Angle);
```

**Parameters:**

Parameter	Description
Angle	Specifies the angle number to play

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STREAM	On failure

**Implementation:**

DVDAngleSelect is wrapped on Sigma Designs' AngleChange function.

**Pseudocode:**

```
U32 VMW_DVD_AngleSelect(U32 Angle)
{
    return AngleChange(Angle);
}
```

**Also see:**

None

### 3.3 VMW\_DVD\_AUDIOSTREAMSELECT

**Description:**

Select the specified stream for playback.

**Syntax:**

```
U32 VMW_DVD_AudioStreamSelect(U32 stream);
```

**Parameters:**

Parameter	Description
stream	Stream number in the available audio stream (1 to 32)

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STREAM	On failure

**Implementation:**

This function is wrapped around Sigma Designs' AudioStreamSelect function.

**Pseudocode:**

```
U32 VMW_DVD_AudioStreamSelect(U32 stream)
{
    return AudioStreamChange(stream);
}
```

**Also see:**

None

### 3.4 VMW\_DVD\_BACKWARDSCAN

**Description:**

Scan the DVD at the specified speed.

**Syntax:**

```
U32 VMW_DVD_BackwardScan(float speed, int fast_slow);
```

**Parameters:**

Parameter	Description
speed	Speed at which to scan and play
fast_slow	0: Backward scanning is at a speed slower than the normal speed 1: Backward scanning is at a higher than normal speed.

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_SCAN	On failure

**Implementation:**

This function is directly mapped to Sigma Designs' API. The DVD\_State is updated as REWIND.

**Pseudocode:**

```
U32 VMW_DVD_BackwardScan(float speed, int fast_slow)
{
    if (!BackwardScan((int)speed, fast_slow))
    {
        dvd_info.dvd_state= REWIND;
        return VMW_OK;
    }
    else
    {
        return VMW_DVDE_SCAN;
    }
}
```

**Also see:**

VMW\_DVD\_ForwardScan

### 3.5 VMW\_DVD\_BUTTONSELECT

**Description:**

Select and activate a given button (or an equivalent mouse click position).

**Syntax:**

```
U32 VMW_DVD_ButtonSelect(U32 param);
```

**Parameters:**

Parameter	Description
param	Represents the button to activate

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_BUTTON	On failure

**Implementation:**

This function is wrapped around Sigma Designs' button select functions.

**Pseudocode:**

```
U32 VMW_DVD_ButtonSelect(U32 param)
{
    switch(param)
    {
        case DVD_UpperButtonSelect:
            if (!UpperButtonSelect()) return VMW_OK;
            break;
        case DVD_LowerButtonSelect:
            if (!LowerButtonSelect()) return VMW_OK;
            break;
        case DVD_LeftButtonSelect:
            if (!LeftButtonSelect()) return VMW_OK;
            break;
        case DVD_RightButtonSelect:
            if (!RightButtonSelect()) return VMW_OK;
            break;
        default:
            break;
    }
    return VMW_DVDE_BUTTON;
}
```

**Also see:**

None

### 3.6 VMW\_DVD\_CALLBACK

**Description:**

Captures the event notifications from the driver.

**Syntax:**

```
DWORD VMW_DVD_Callback(DWORD dwContext, DWORD dwMsg, DWORD dwValue);
```

**Parameters:**

None

**Returns:**

Zero

**Implementation:**

This function is registered while opening the driver. Based on the driver event, the structure dvd\_info is updated. dvd\_info is a global structure, that holds the information such as the current title number, chapter number, current time and the state of the player.

```
typedef struct
{
    U32 chapter_no;
    U32 title_no;
    U32 time;
    DVD_State_t dvd_state;
} DVD_Info_t;
DVD_Info_t dvd_info ;
```

**Pseudocode:**

```
DWORD VMW_DVD_Callback( DWORD dwContext, DWORD dwMsg , DWORD dwValue)
{
    switch(dwMsg)
    {
        case FMPM_DVD_TITLE_CHANGE :
            dvd_info.title_no = dwValue;
            break;
        case FMPM_DVD_CURRENT_TIME:
            dvd_info.time=dwValue;
            break;
        case FMPM_DVD_PLAYBACK_STOPPED:
            dvd_info.dvd_state= STOP;
            break;
        case FMPM_DVD_PROGRAM_START:
            dvd_info.chapter_no =dwValue;
            break;
        case FMPM_DVD_ANGLES_BLOCK:
        case FMPM_DVD_AUDIO_STREAM_CHANGE:
        case FMPM_DVD_BUTTONS_CHANGE:
        case FMPM_DVD_BUTTON_CHANGE:
        case FMPM_DVD_DOMAIN_CHANGE:
        case FMPM_DVD_NO_FP_PGC:
        case FMPM_DVD_PARENTAL_LEVEL_CHANGE:
        case FMPM_DVD_STILL_OFF:
        case FMPM_DVD_STILL_ON:
        case FMPM_DVD_SUBPICTURE_STREAM_CHANGE:
        case FMPM_DVD_VALID_UOPS_CHANGE:
        default:
            break;
    }
    return 0;
}
```

**Also see:**

None

### 3.7 VMW\_DVD\_CHAPTERPLAY

**Description:**

Play the DVD from the specified title and chapter.

**Syntax:**

U32 VMW\_DVD\_ChapterPlay(U32 title, U32 chapter);

**Parameters:**

Parameter	Description
title	Title of the DVD to be played (title is between 1 and 99)
chapter	Chapter to play under selected title (chapter is between 1 and 999)

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_CHAPTER	On failure

**Implementation:**

VMW\_DVD\_ChapterPlay function is wrapped around Sigma Designs' ChapterPlay function.

**Pseudocode:**

```
U32 VMW_DVD_ChapterPlay(U32 title,U32 chapter)
{
    return ChapterPlay(title,chapter);
}
```

**Also see:**

TitlePlay

### 3.8 VMW\_DVD\_CLOSE

**Description:**

Close the device driver and perform any necessary clean-up of the driver or device.

**Syntax:**

```
U32 VMW_DVD_Close();
```

**Parameters:**

None

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_CLOSE	On failure

**Implementation:**

This function closes DVD disc access and MPEG2 decoder, and unloads the driver. The DVD\_State is updated as CLOSE.

**Pseudocode:**

```
U32 VMW_DVD_Close()
{
    if(!FMPStop())
    if(!FMPClose())
    if(!MPEGDriverUnload())
    {
        dvd_info.dvd_state=CLOSE;
    }
    return VMW_DVDE_CLOSE;
}
```

**Also see:**

VMW\_DVD\_Open

### 3.9 VMW\_DVD\_FORWARDSCAN

**Description:**

Scan the DVD at the specified speed, because if parameter speed is 0-1, it retains normal motion (refer to parameter description).

**Syntax:**

U32 VMW\_DVD\_ForwardScan(float speed, int fast\_slow);

**Parameters:**

Parameter	Description
speed	Speed at which to scan and play
fast_slow	0: Forward scanning is at a speed slower than the normal speed 1: Forward scanning is at a higher than normal speed

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_SCAN	On failure

**Implementation:**

This function is wrapped around Sigma Designs' ForwardScan function. The DVD\_State is updated as FF or SLOW depending on the value of fast\_slow.

**Pseudocode:**

```
U32 VMW_DVD_ForwardScan(float speed, int fast_slow)
{
    if (!ForwardScan((int)speed, fast_slow))
    {
        if(fast_slow == TRUE ) dvd_info.dvd_state= FF;
        else dvd_info.dvd_state=SLOW;
        return VMW_OK;
    }
    else
    {
        return VMW_DVDE_SCAN;
    }
}
```

**Also see:**

VMW\_DVD\_BackwardScan

### 3.10 VMW\_DVD\_GETCURRENTCHAPTER

**Description:**

This function returns the number of the chapter currently being played.

**Syntax:**

```
U32 VMW_DVD_GetCurrentChapter();
```

**Parameters:**

None.

**Returns:**

Returns the current chapter.

**Implementation:**

Returns the chapter number from the structure `dvd_info`. The structure `dvd_info` is updated based on the driver event.

**Pseudocode:**

```
U32 VMW_DVD_GetCurrentChapter()  
{  
    return dvd_info.chapter_no;  
}
```

**Also see:**

VMW\_DVD\_Callback

### 3.11 VMW\_DVD\_GETCURRENTPLAYSTATE

**Description:**

Returns the current state of the DVD player. The state of the player is maintained by the dvd\_info structure and is updated whenever the state transition happens.

**Syntax:**

```
U32 VMW_DVD_GetCurrentPlayState();
```

**Parameters:**

None

**Returns:**

Returns the current play state. The following is a list of playback states: OPEN, PLAY, FF, SLOW, REWIND, PAUSE, STOP, CLOSE.

**Implementation:**

This function returns the state from the structure dvd\_info. The field dvd\_state in the structure dvd\_info holds the state information as follows.

```
typedef enum
{
    UNDEFINED = 200,           // Initial State.
    OPEN,                     //The driver has been opened.
    PLAY,                     //The dvd is being played at normal speed.
    FF,                       //The dvd is being played at faster speed.
    SLOW,                     //The dvd is being played at slow speed.
    REWIND,                   //The dvd is being played reverse.
    PAUSE,                    //Play back has been paused
    STOP,                     //Play back has been stopped
    CLOSE,                    //The Driver has been Closed.
} DVD_State_t;
DVD_State_t dvd_state;
```

**Pseudocode:**

```
U32 VMW_DVD_GetCurrentPlayState()
{
    return dvd_info.dvd_state;
}
```

**Also see:**

VMW\_DVD\_Open, VMW\_DVD\_Close, VMW\_DVD\_ForewardScan, VMW\_DVD\_BackwardScan, VMW\_DVD\_Play, VMW\_DVD\_PauseON, VMW\_DVD\_PauseOff, VMW\_DVD\_Stop

### 3.12 VMW\_DVD\_GETCURRENTTIME

**Description:**

This function returns the time stamp of the chapter currently being played.

**Syntax:**

```
U32 VMW_DVD_GetCurrentTime();
```

**Parameters:**

None

**Returns:**

Returns the current playback time in seconds.

**Implementation:**

This function returns the time from the structure `dvd_info`. Based on the driver event, the structure `dvd_info` is updated.

**Pseudocode:**

```
U32 VMW_DVD_GetCurrentTime()  
{  
    return dvd_info.time;  
}
```

**Also see:**

`VMW_DVD_Callback`

### 3.13 VMW\_DVD\_GETCURRENTTITLE

**Description:**

This function returns the number of the title currently being played.

**Syntax:**

```
U32 VMW_DVD_GetCurrentTitle();
```

**Parameters:**

None.

**Returns:**

Returns the title.

**Implementation:**

This function is returns the title number from the structure dvd\_info. Based on the driver event, the structure dvd\_info is updated.

**Pseudocode:**

```
U32 VMW_DVD_GetCurrentTitle()  
{  
    return dvd_info.title;  
}
```

**Also see:**

VMW\_DVD\_Callback

### 3.14 VMW\_DVD\_KARAOKE

**Description:**

Set the audibility of the different karaoke tracks.

**Syntax:**

```
U32 VMW_DVD_Karaoke(U32 mixingMode);
```

**Parameters:**

Parameter	Description
mixingMode	Sets the source to mix into left/right channels according to: Bit 0: 0: Do not do the mixing 1: Do the mixing Bit 2: Mix Ch2 to Ch1 (melody to Right) Bit 3: Mix Ch3 to Ch1 (Voice1 to Right) Bit 4: Mix Ch4 to Ch1 (Voice2 to Right) Bit 10: Mix Ch2 to Ch0 (Melody to Left) Bit 11: Mix Ch3 to Ch0 (Voice1 to Left) Bit 12: Mix Ch3 to Ch1 (Voice2 to Left)

**Returns:**

Return	Description
VMW_OK	On success
Non-zero on failure	

**Implementation:**

This function is wrapped around Sigma Designs' KaraokeAudioPresentationMode function.

**Pseudocode:**

```
U32 VMW_DVD_karaoke(U32 mixingMode)
{
    return KaraokeAudioPresentationMode (mixingMode);
}
```

**Also see:**

None

**3.15 VMW\_DVD\_LANGUAGESelect****Description:**

Selects the language for the system menu according to the specified language code.

**Syntax:**

```
U32 VMW_DVD_LanguageSelect(char* planguagecode);
```

**Parameters:**

Parameter	Description
planguagecode	A two-character ISO 639-2 language code is used to select the language code for menus.

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STREAM	On failure

**Implementation:**

This function is wrapped around Sigma Designs' MenuLanguageSelect function.

**Pseudocode:**

```
U32 VMW_DVD_Languageselect(char* pLanguageCode)
{
    return (MenuLanguageSelect(*pLanguageCode));
}
```

**Also see:**

None

### 3.16 VMW\_DVD\_OPEN

**Description:**

Open the driver DVD playback.

**Syntax:**

```
U32 VMW_DVD_Open();
```

**Parameters:**

None

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_OPEN	On failure

**Implementation:**

DVDOpenStruct structure is initialized. The driver is opened by calling FMPOpenDiscPlayback. The DVD\_State is updated as OPEN (see VMW\_DVD\_Callback).

**Pseudocode:**

```
U32 VMW_DVD_Open()
{
    FMP_OPENSTRUCT DVDOpenStruct;
    if (!MPEGDriverEntry(5)) //success
    {
        //Initializing DVDOpenStruct
        if(!FMPOpenDiscPlayback (&DVDOpenStruct)) //success
        {
            dvd_info.dvd_state = OPEN;
            return VMW_OK;
        }
    }
    return VMW_DVDE_OPEN; //error
}
```

**Also see:**

VMW\_DVD\_Close

### 3.17 VMW\_DVD\_PAUSEON

**Description:**

Pause a playing DVD stream.

**Syntax:**

U32 VMW\_DVD\_PauseOn(void);

**Parameters:**

None

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_PAUSE	On failure

**Implementation:**

This function is wrapped around Sigma Designs' FMPPause function. The stream will be paused by the FMPPause() function, and the decoder will stop decoding the stream and playback. The DVD\_State is updated as PAUSE.

**Pseudocode:**

```
U32 VMW_DVD_PauseOn(void)
{
    if (!FMPPause())
    {
        dvd_info.dvd_state = PAUSE;
        return VMW_OK;
    }
    else
    {
        return VMW_DVDE_PAUSE;
    }
}
```

**Note:** This will pause the audio and video decoder. The source will send the stream before pause and it will not be decoded.

**Also see:**

VMW\_DVD\_PauseOff

### 3.18 VMW\_DVD\_PAUSEOFF

**Description:**

Pause a playing DVD stream.

**Syntax:**

```
U32 VMW_DVD_PauseOff(void);
```

**Parameters:**

None

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_PAUSE	On failure

**Implementation:**

The stream will be played by the FMPPPlay() function. The decoder decodes the stream and starts playback. The DVD\_State is updated as PLAY.

**Pseudocode:**

```
U32 VMW_DVD_PauseOff(void)
{
    if(!FMPPPlay())
    {
        dvd_info.dvd_state= PLAY;
        return VMW_OK;
    }
    else
    {
        return VMW_DVDE_PAUSE;
    }
}
```

**Also see:**

VMW\_DVD\_PauseOn

VMW\_DVD\_Play

**3.19 VMW\_DVD\_PLAY****Description:**

Puts the driver in Play mode.

**Syntax:**

```
U32 VMW_DVD_Play();
```

**Parameters:**

None

**Returns:**

Return	Description
VMW_OK	On success
	VMW_DVDE_PLAY

**Implementation:**

This function puts the driver in Play mode by calling FMPPPlay(). The DVD\_State is updated as PLAY.

**Pseudocode:**

```
U32 VMW_DVD_Play()
{
    if(!FMPPPlay())
    {
        dvd_info.dvd_state = PLAY;
        return VMW_OK;
    }
    else
    {
        return VMW_DVDE_PLAY;
    }
}
```

**Also see:**

VMW\_DVD\_Close, VMW\_DVD\_PauseOn, VMW\_DVD\_PauseOff

### 3.20 VMW\_DVD\_SEARCH

**Description:**

Stop playing the DVD and enter the search mode.

**Syntax:**

U32 VMW\_DVD\_Search(U32 searchMode,U32 searchParam);

**Parameters:**

Parameter	Description
SearchMode	This flag indicates one of the following search modes: DVD_CHAPTER_SRCH DVD_TIME_SRCH DVD_GOUP_SRCH DVD_TOP_SRCH DVD_NEXT_SRCH DVD_PREV_SRCH

The SearchParam is required only for the parameter SearchMode set to DVD\_TIME\_SRCH or DVD\_CHAPTER\_SRCH.

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_INSEARCH	On failure

**Implementation:**

The DVD search functions are combined to implement the above search options. The Sigma Designs' TimeSearch function's parameter is defined as STRING.

**Pseudocode:**

```
VMW_DVD_Search(U32 searchMode,U32*searchParam)
{
  switch(searchMode){
    case DVD_TIME_SRCH:
      if (TimeSearch(searchParam)) return VMW_OK;
    case DVD_CHAPTER_SRCH:
      if (!ChapterSearch(*searchParam)) return VMW_OK;
      break;
      //specified PTT numbers within the same title.
    case DVD_GOUP_SRCH:
      if (!GoUp())return VMW_OK;
      break;
      //Stop the execution of current program chain and play the GoUp
    case DVD_TOP_SRCH:
      **
      if (!TopSearch()) return VMW_OK;
      break;
    case DVD_PREV_SRCH:
      if (PrevPGSearch())return VMW_OK;
      break;
    case DVD_NEXT_SRCH:
      if (NextPGSearch())return VMW_OK;
      break;
    default:
      break;
  }
  return VMW_DVDE_INSEARCH;
}
```

**Also see:**

None

**3.21 VMW\_DVD\_SETPARENTALCONTROL****Description:**

Set the country and level of parental management to be applied.

**Syntax:**

```
U32 VMW_DVD_SetParentalControl(U32 country, U32 level);
```

**Parameters:**

Parameter	Description
country	Two-character country code, according to ISO 3166, to be used for parental management system
level	Specifies the level (1-8, 0 = disabled) for parental management system

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STREAM	On failure

**Implementation:**

This function is used to set parental country select and parental level select.

**Pseudocode:**

```
U32 VMW_DVD_SetParentalControl(U32 country, U32 level)
{
    if (!ParentalCountrySelect(country)) // success
        return (ParentalLevelSelect(level));
    else
        return VMW_DVDE_STREAM;
}
```

**Also see:**

None

### 3.22 VMW\_DVD\_SETVIDEOPRESENTATIONMODE

**Description:**

Set the video presentation mode.

**Syntax:**

U32 VMW\_DVD\_SetVideoPresentationMode(U32 aspectRatio, U32 displaymode);

**Parameters:**

Parameter	Description
aspectRatio	Sets the aspect ratio according to: 0: Aspect ratio of 4:3 1: Aspect ratio of 16:9
displaymode	Sets the display mode according to: 0: Normal (4:3 or 16:9) 1: Pan-scan 2: Letterbox

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STREAM	On failure

**Implementation:**

This function is wrapped around Sigma Design's VideoPresentationModeChange function.

**Pseudocode:**

```
U32 VMW_DVD_SetVideoPresentationMode(U32 aspectRatio, U32 displaymode)
{
    return VideoPresentationModeChange(aspectRatio, displayMode);
}
```

**Also see:**

None

### 3.23 VMW\_DVD\_STOP

**Description:**

Stop playing the DVD.

**Syntax:**

U32 VMW\_DVD\_Stop(void);

**Parameters:**

None

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STOP	On failure

**Implementation:**

This function is wrapped around Sigma Designs' FMPStop function. The DVD\_State is updated as STOP.

**Pseudocode:**

```
U32 VMW_DVD_Stop(void)
{
    if(!FMPStop())
    {
        dvd_state = dvd_info.dvd_state=STOP;
        return VMW_VMW_OK;
    }
    return VMW_DVDE_STOP;
}
```

**Also see:**

VMW\_DVD\_Play, VMW\_DVD\_PauseOff

### 3.24 VMW\_DVD\_SUBPICTURESTREAMSELECT

**Description:**

Select the specified subpicture stream for playback.

**Syntax:**

```
U32 VMW_DVD_SubPictureStreamSelect(U32 subPicStream, U32 setDisplay);
```

**Parameters:**

Parameter	Description
subPicStream	Stream number in the available SubPicture stream
setDisplay	0: Sets the substream display off 1: Sets the substream display on

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_STREAM	On failure

**Implementation:**

VMW\_DVD\_SubPictureStreamSelect is wrapped on Sigma Design's SubPictureStreamChange function.

**Pseudocode:**

```
U32 VMW_DVD_SubPictureStreamSelect(U32 subPicStream, U32 setDisplay)
{
    return SubPictureStreamChange(subPicStream, setDisplay);
}
```

**Also see:**

None

### 3.25 VMW\_DVD\_TITLEPLAY

**Description:**

Play the DVD from the specified title.

**Syntax:**

```
U32 VMW_DVD_TitlePlay(U32 title);
```

**Parameters:**

Parameter	Description
title	Title of the DVD to be played (title is between 1 and 99)

**Returns:**

Return	Description
VMW_OK	On success
VMW_DVDE_TITLENUMBER	On failure

**Implementation:**

VMW\_DVD\_TitlePlay function is wrapped around Sigma Designs' TitlePlay function.

**Pseudocode:**

```
U32 VMW_DVD_TitlePlay(U32 title)
{
    U32 maxTitle;
    maxTitle = FMPI_Get(FMPI_TITLES_AVAILABLE);
    if(maxTitle== 0xffffffff)
        return VMW_DVDE_TITLENUMBER;
    if (title >= maxTitle) title = maxTitle;
    if (!TitlePlay(title))
    {
        return VMW_OK;
    }
    else
    {
        return VMW_DVDE_TITLENUMBER;
    }
}
```

**Also see:**

VMW\_DVD\_ChapterPlay

## 4.0 Error Codes

**Table 4-1. DVD API Function Error Codes**

<b>Error Code</b>	<b>Description</b>
VMW_DVDE_TITLENUMBER	Error returned for invalid specification of title numbers
VMW_DVDE_CHAPTER	Error returned for invalid specification of chapter numbers
VMW_DVDE_CLOSE	Error returned during closing of DVD
VMW_DVDE_INSEARCH	Error returned during invalid searches
VMW_DVDE_MENU	Error returned during activating menus
VMW_DVDE_OPEN	Error returned during opening of DVD
VMW_DVDE_PAUSE	Error returned during pause
VMW_DVDE_SCAN	Error returned during forward or backward scanning
VMW_DVDE_STOP	Error returned during stop
VMW_DVDE_STREAM	Error returned during streaming

## Appendix A Support Documentation

### A.1 REVISION HISTORY

This document is a report of the revision/creation process. Any revisions (i.e., additions, deletions, parameter corrections, etc.) are recorded in the table(s) below.

**Table A-1. Revision History**

Revision # (PDF Date)	Revisions / Comments
1.0 (7/06/01)	Release 1.0
1.1 (08/17/01)	Rewrote all pseudocode.

## LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.



**National Semiconductor Corporation Americas**  
Email: new.feedback@nsc.com

**National Semiconductor Europe**  
Fax: +49 (0) 180-530 85 86  
Email: europe.support@nsc.com  
Deutsch Tel: +49 (0) 69 9508 6208  
English Tel: +44 (0) 870 24 0 2171  
Français Tel: +33 (0) 1 41 91 87 90

**National Semiconductor Asia Pacific Customer Response Group**  
Tel: 65-2544466  
Fax: 65-2504466  
Email: ap.support@nsc.com

**National Semiconductor Japan Ltd.**  
Tel: 81-3-5639-7560  
Fax: 81-3-5639-7507  
Email: nsj.crc@jksmtp.nsc.com

[www.national.com](http://www.national.com)