Standard for RealTek DVD Recordable

Date: 4/28/2012

DVD-VENUS 2D Graphics API

RealTek specification on DVD Recordable Technology

Specification for DVD-VENUS: Venus 2D Graphics Application Interface

Note
This document is subject to review.

Chen Ma
chen.ma@realcomtec.com

Version	Date	Author
0.0.1	10/20/04	Chen Ma

Change List	Version	Description

Table of Contents

1 List of Figures

2 Introduction

2.1 Purpose

Describe the Visual User Interface design APIs. This document serve as basic drawing API for Venus system. For Window API, please refer to Venus Window Programming API. The 2D Graphics API complies two part, **Direct Graphics** and **GDI** API. Direct Graphics is the hardware accelerated Graphic APIs supported by Venus system. The GDI API is the software library that supports basic drawing functions and objects.

2.2 Scope

This document covers the introduction and the application interfaces of Direct Graphics. It also mention about how to setup and use the GDI library.

2.3 References

- Operating System Abstraction Layer (OSAL)

3 Deployment Diagram

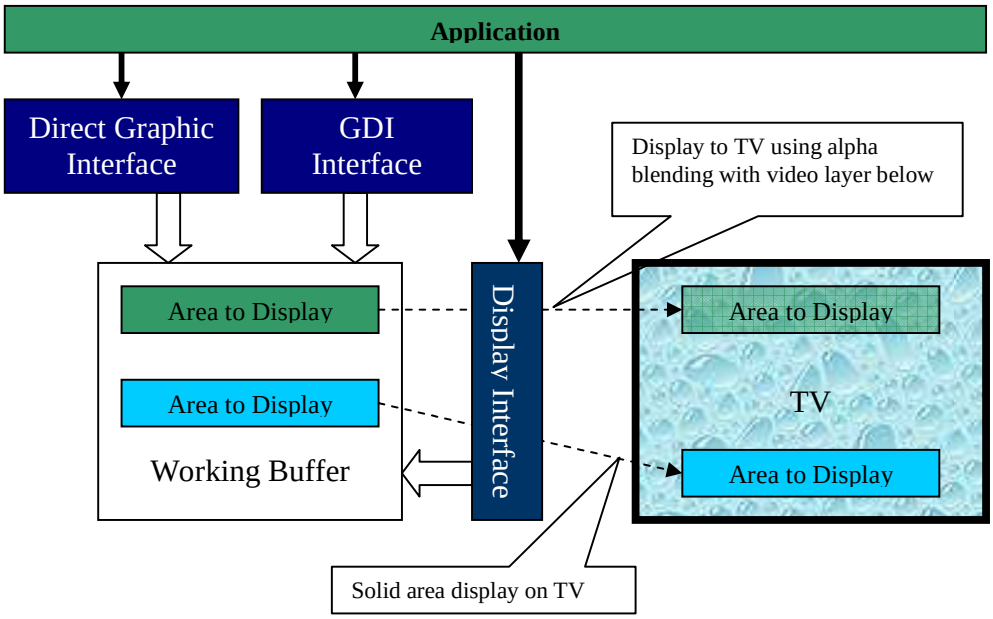

Figure 1 The Graphic Interface uncovered

Figure 1 demonstrates the DVD+VR recording flow under StreamClass architecture. The Flow Manager is a class that serves as a gateway between Filters and Application.

4 Direct Graphics Interfaces

4.1 DG_CreateSurface

DG_CreateSurface method creates a drawing surface.

Syntax

HANDLE DG_CreateSurface(SURFACEDESC *pDesc)

Parameters

pDesc

Pointer to a **SURFACEDESC** structure that contains the information necessary to create the drawing object.

Return

The handle to the created surface.

Remarks

The **lPitch** and **lpSurface** members are output values when calling the **DG_GetSurfaceDesc** method. When creating surfaces from existing memory or updating surface characteristics, these members are input values that describe the pitch and location of memory allocated by the calling application for use by 2D graphics library. The library does not attempt to manage or free memory allocated by the application.

4.2 DG_GetSurfaceDesc

DG_GetSurfaceDesc method retrieves the descriptor of drawing surface.

Syntax

int DG_GetSurfaceDesc(HANDLE handle, SURFACEDESC *pDesc)

Parameters

handle

Handle to the drawing surface.

pDesc

Pointer to a **SURFACEDESC** structure that will retrieve the information for the drawing object.

4.3 DG_AlphaBlt

```
DG_AlphaBlt(HANDLE handleDes,
            int nXDest,
            int nYDest,
            int nWidth,
            int nHeight,
            HANDLE handleSrc,
            int srcColor,
            int nXSrc,
            int nYSrc,
            ALPHA_MODE alphaMode,
            BYTE srcA,
            BYTE desA,
            bool isForceDestAlpha,
            BYTE forcedDesA,
            KEY_MODE colorKeyMode,
            unsigned long colorKey);
```

Parameters

handleDes

[in] Handle to the destination drawing surface.

nXDest

[in] Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

nYDest

[in] Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

nWidth

[in] Specifies the logical width of the source and destination rectangles.

nHeight

[in] Specifies the logical height of the source and the destination rectangles.

handleSrc

[in] Handle to the source drawing surface. If handleSrc equal to 0, the source is a constant color.

srcColor

[in] This field take effect when handleSrc equal to 0, otherwise this field will be ignored. The srcColor represents the native color value of the source constant color. Function will take lower 8 bits of srcColor if destination surface is a 8 bits color plane. Same concept apply to other color depth destination surface with the lower n bits as the source constant color value.

nXSrc

[in] Specifies the logical x-coordinate of the upper-left corner of the source rectangle.

nYSrc

[in] Specifies the logical y-coordinate of the upper-left corner of the source rectangle.

alphaMode

[in] see ALPHA_MODE.

Choose none if no alpha blending is needed.

If choose constant alpha. Set the alpha value in both srcA and desA.

out = src *srcA + des * desA // Same operation apply to all the channels (RGBA)

If choose source alpha as major, and operate on per pixel alpha. This operation only applicable on 32 bits format.

out = src *srcA_pixel + des * (1-srcA_pixel) // Same operation apply to all the channels (RGBA)

If choose destination alpha as major, and operate on per pixel alpha. This operation only applicable on 32 bits format.

out = src *(1-desA_pixel) + des * desA_pixel) // Same operation apply to all the channels (RGBA)

srcA

[in] source alpha value.

desA

[in] destination alpha value.

isForcedestAlpha

[in] set to true if want to set the value of destination alpha channel after blt operation to become a constant value.

forcedDesA

[in] This parameter contains the forced destination alpha channel value.

colorKeyMode

[in] see KEY_MODE. Choose between source color key, destination color key or no color key applied. If choose source color key, the source pixels that match the

color key value will be transparent on the destination. On the other hand, if destination key is chose, the destination pixels that match the destination color key will be replaced by corresponding source pixels. When color key is enabled, color key operations suppress alpha operations.

colorKey

[in] Color key value. Depends on color format, alpha channel in 32 bits mode will be ignored when doing the key comparison.

4.4 DG_BitBlt

```
DG_BitBlt(  HANDLE handleDes,
            int nXDest,
            int nYDest,
            int nWidth,
            int nHeight,
            HANDLE handleSrc,
            int nXSrc,
            int nYSrc,
            ROP_CODE ropCode);
```

Parameters

handleDes

[in] Handle to the destination drawing surface.

nXDest

[in] Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

nYDest

[in] Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

nWidth

[in] Specifies the logical width of the source and destination rectangles.

nHeight

[in] Specifies the logical height of the source and the destination rectangles.

handleSrc

[in] Handle to the source drawing surface.

nXSrc

[in] Specifies the logical x-coordinate of the upper-left corner of the source rectangle.

nYSrc

[in] Specifies the logical y-coordinate of the upper-left corner of the source rectangle.

ropCode

[in] Specifies a raster-operation code. These codes define how the color data for the source rectangle is to be combined with the color data for the destination rectangle to achieve the final color.

The following list shows some common raster operation codes:

Value	Description
BLACKNESS	Fills the destination rectangle using the color associated with index 0 in the physical palette. (This color is black for the default physical palette.)
DSTINVERT	Inverts the destination rectangle.
MERGECOPY	Merges the colors of the source rectangle with the specified pattern by using the Boolean AND operator.
MERGEPAINT	Merges the colors of the inverted source rectangle with the colors of the destination rectangle by using the Boolean OR operator.
NOTSRCCOPY	Copies the inverted source rectangle to the destination.
NOTSRCERASE	Combines the colors of the source and destination rectangles by using the Boolean OR operator and then inverts the resultant color.
PATCOPY	Copies the specified pattern into the destination bitmap.
PATINVERT	Combines the colors of the specified pattern with the colors of the destination rectangle by using the Boolean XOR operator.
PATPAINT	Combines the colors of the pattern with the colors of the inverted source rectangle by using the Boolean OR operator. The result of this operation is combined with the colors of the destination rectangle by using the Boolean OR operator.
SRCAND	Combines the colors of the source and destination rectangles by using the Boolean AND operator.
SRCCOPY	Copies the source rectangle directly to the destination rectangle.
SRCERASE	Combines the inverted colors of the destination rectangle with the colors of the source rectangle by using the Boolean AND operator.
SRCINVERT	Combines the colors of the source and destination rectangles by using the Boolean XOR operator.
SRCPAINT	Combines the colors of the source and destination rectangles by using the Boolean OR operator.
WHITENESS	Fills the destination rectangle using the color associated with index 1 in the physical palette. This color is white for the default physical palette.

4.5 DG_Lock

```
DG_Lock(    HANDLE handleDes,
            int nXDest,
            int nYDest,
```

```
            int nWidth,
            int nHeight,
            int dwFlags,
)
```

Parameters

handleDes
[in] Handle to the destination drawing surface.
nXDest
[in] Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.
nYDest
[in] Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.
nWidth
[in] Specifies the logical width of the source and destination rectangles.
nHeight
[in] Specifies the logical height of the source and the destination rectangles.
dwFlags
[in] Reserved

Remark
Lock the surface for direct memory access. A surface can be locked once before calling the unlock. The Lock function will guarantee the completion of hardware acceleration before the return of the Lock function. All functions prefix with DG_ are hardware accelerated functions, which are executed asynchronously to the main program. In order to do direct access to the surface, one must make sure the previously issued DG functions have been executed, to prevent overwritten by DG function again.

4.6 DG_Unlock

DG_Unlock(**HANDLE** *handleDes*)

Parameters

handleDes
[in] Handle to the destination drawing surface.

Remark
Unlock the surface. A surface can be locked once before calling the unlock.

4.7 DG_DrawPixel

```
DG_ DrawPixel (HANDLE handleDes,
               int nXDest,
               int nYDest,
               int srcColor
              )
```

Parameters

handleDes

[in] Handle to the destination drawing surface.

nXDest

[in] Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

nYDest

[in] Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

srcColor

[in] The srcColor represents the native color value of the source constant color. Function will take lower 8 bits of srcColor if destination surface is a 8 bits color plane. Same concept apply to other color depth destination surface with the lower n bits as the source constant color value.

Remark

The DG_DrawPixel will queue the drawing command into the hardware queue. Depends on the queue size, the DG_DrawPixel might not be the best way to draw thousands of pixels. For large bit map drawing, it is prefer to call DG_Lock, then directly access the surface using direct memory access.

4.8 Drawing Lines

Use DG_DrawRectangle to draw line. While drawing horizontal line, set nHeight to 1. When drawing vertical lines, set nWidth to 1. Only horizontal line and vertical line are supported by the hardware. Use GDI functions to draw other slop lines.

4.9 Drawing Rectangle

A wrapper function is provided for simple opaque rectangle drawing on the destination surface.

```
DG_DrawRectangle( HANDLE handleDes,
                  int nXDest,
                  int nYDest,
                  int nWidth,
                  int nHeight,
                  int srcColor )
```

We use **DG_AlphaBlt** to implement the DrawRectangle by setting *srcHandle* to 0, and set the source color to *srcColor.* See **DG_AlphaBlt** for details. To enable alpha feature or color key feature, use **DG_AlphaBlt** directly.

5 Display Interface

5.1 DP_ GetDisplayHandle

```
HANDLE DP_GetDisplayHandle ( );
```

Return
NULL if no more display hardware is available.

Remarks
You need to get a valid handle before subsequent call to display functions. You can use the same handle for multiple display function calling.

5.2 DP_ DisplayArea

```
DP_DisplayArea ( int areaHandle,
                 int nXDest,
                 int nYDest,
                 int nWidth,
                 int nHeight,
                 HANDLE handle,
                 int nXSrc,
                 int nYSrc,
                 ALPHA_MODE  alphaMode,
                 BYTE srcA,
                 KEY_MODE colorKeyMode,
                 unsigned long colorKey )
```

Parameters
displayHandle
[in] Specifies the handle to display area;
nXDest
[in] Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.
nYDest
[in] Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.
nWidth
[in] Specifies the logical width of the source and destination rectangles.
nHeight
[in] Specifies the logical height of the source and the destination rectangles.
handle
[in] Handle to the working buffer, which is the source.
nXSrc
[in] Specifies the logical x-coordinate of the upper-left corner of the source rectangle.
nYSrc

[in] Specifies the logical y-coordinate of the upper-left corner of the source rectangle.

alphaMode

[in] see ALPHA_MODE. Only `Alpha_None, Alpha_Constant` and `Alpha_SrcMajor` are supported.
Choose none if no alpha blending is needed.
If choose constant alpha. Set the alpha value in srcA.

out = src *srcA + des * (1- srcA) // Same operation apply to all the channels (RGBA)

If choose source alpha as major, and operate on per pixel alpha. This operation only applicable on 32 bits format.

out = src *srcA_pixel + des * (1-srcA_pixel) // Same operation apply to all the channels (RGBA)

srcA

[in] source alpha value.

colorKeyMode

[in] see KEY_MODE. Only `ColorKey_Src` and `CloroKey_None` are supported.

colorKey

[in] Color key value.

5.3 DP_ ReleaseDisplayHandle

DP_ReleaseDisplayHandle (**`int`** *`areaHandle`*);

Parameters

displayHandle

[in] Specifies the handle to display area;

6 GDI Interface

6.1 Button

We support generic button as a C++ object.

7 Messaging System

8 Data Structure Design

8.1 Object handle: HANDLE

```
typedef void*  HANDLE;
```

8.2 Surface Descriptor: SURFACEDESC

#DEFINE DWORD UNSIGNED LONG

```
typedef struct _SURFACEDESC {
    DWORD            dwSize;
    DWORD            dwFlags;
    DWORD            dwHeight;
    DWORD            dwWidth;
    LONG             lPitch;
    DWORD            dwBackBufferCount;
    DWORD            dwRefreshRate;
    DWORD            dwAlphaBitDepth;
    VOID*            lpSurface;
    PIXEL_FORMAT      pixelFormat;
} SURFACEDESC;
```

Members

dwSize

Size of the structure, in bytes. This member must be initialized before the structure is used.

dwHeight and **dwWidth**

Dimensions of the surface to be created, in pixels.

lPitch

Distance, in bytes, to the start of next line.

dwRefreshRate

Refresh rate (used when the display mode is described). The value of 0 indicates an adapter default.

dwBackBufferCount

Not used.

lpSurface

Address of the associated surface memory. It has to be 4 page boundary. The size of one memory page depends on SDRAM type. Use getSdramPageSize() to get the size;

pixelFormat

8 bits, 16 bits or 32 bits. See **PIXEL_FORMAT.**

9 ENUM definitions

9.1.1 PIXEL_FORMAT

```
typedef enum _PIXELFORMAT
{
    Format_8,            //332
    Format_16,           //565
    Format_32,           //A888
    Format_32_888A       //888A
} PIXEL_FORMAT;
```

9.1.2 ALPHA_MODE

```
typedef enum _ALPHAMODE
{
    Alpha_None
```

```
    Alpha_Constant,
    Alpha_SrcMajor,
    Alpha_DecMajor,
} ALPHA_MODE;
```

9.1.3 COLOR_KEY_MODE

```
typedef enum _COLORKEYMODE
{
    ColorKey_Src,
    ColorKey_Des,
    CloroKey_None,
} COLOR_KEY_MODE;
```

10 Real-Time Design

Use Direct Graphics interface will significantly increase the performance of the system compare to use the GDI API.

User Interface Design

< Refer to User Interface material in the SRS and supplement with any design considerations not mentioned there. You should discuss the expected effectiveness of your design. >

11 Help System Design

12 Index

< generate here >