GM8136

THINK2DGE

User Guide
Rev.: 1.0
Issue Date: August 2014

REVISION HISTORY

GM8136 THINK2DGE User Guide

Date	Rev.	From	To
Aug. 2014	1.0	-	Original

Printed in Taiwan 2014

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This chapter contains the following sections:

1.1 Introduction

The 2D computer Graphics accelerator Engine (2D GE) is designed to improve the performance of the computer GUI functions, such as BLTs and Drawing. A pixel is the smallest addressable screen element defined in Microsoft Windows, and lines and pictures are composed of a variety of pixels. 2D GE is used to off-load the CPU overhead and speed up the graphic performance in moving the pixels and drawing.

1.2 Suggested Readers

Software designers of the applications and drivers are recommended to read this document.

1.3 Features

- 2D drawing engine
  - Pixel drawing
  - Line drawing
  - Filled rectangles
- Color formats (RGBA8888, ARGB8888, RGB565, RGBA5551, and RGBA4444)
- Alpha blending
- Blitter
  - Moves raster images of high-performance DMA blitter into memory
  - Image format color conversion on-the-fly
  - Stretches x and y axes
  - Source color keying
  - Supports rotations for Mirror-X and Mirror-Y in 90, 180, and 270 degrees

1.4 Included Files

As shown in Figure 1-1, the THINK2DGE driver is provided to set the parameter for the hardware and trigger the hardware drawing. The common library is the interface between the application and the THINK2DGE driver. think2d_lib.*h* should be included in the application. think2d_lib.*h* defines the constant and user-defined data type, and declares the function calls in the application provided in think2dge_lib.*o*.

Figure 1-1. Software Architecture

Table 1-1. File List

File Name	Description
think2d_lib.h	Header file that should be included in the application layer
think2dge_lib.o	think2dge_lib.o sends the draw and blit commands to the THINK2DGE driver.
think2d.ko	think2d.ko is the driver that programs the hardware.

Chapter 2

Software Architecture

This chapter contains the following section:

- 2.1 Software Design Architecture

2.1 Software Design Architecture

A draw command draws one pixel or one line, or fills one rectangle. If users want to draw two lines, two draw commands should be issued. For the latter case, the 2D GE hardware supports the command list table, which allows users to add multiple draw commands to the command list table. Users can call the emit function and all commands stored in the command list table will be executed.

2.1.1 Share Memory Region

In a driver, a memory region will be created to store the draw commands, which will be added into the command list table. When the emit function is called by users, all commands will be stored in the share memory, then the driver will be triggered to start sending commands. When the driver gets an interrupt from the hardware, it means that all commands in the command list table are executed.

The size of the command list table is 4080 entries, which means that the command list table can store 2040 commands. One entry is the command type and the other entry is the command parameter.

2.1.2 Open and Close Function

Before using 2D GE to draw pictures, users should open the channel first. Think2dge_Lib_Open () is used to open 2D GE. Multiple open operations are allowed. Inside Think2dge_Lib_Open (), users can see that the following two device nodes are opened:

```
lcd_fd = open("/dev/fb1", O_RDWR);
2DGE_fd = open("/dev/think2d", O_RDWR);
```

When Think2dge_Lib_Open () is called, users must pass the RGB type to the open function. In order to save the bandwidth, 16bpp will be used for a plane, such as RGB565. The RGB565 plane is the mostly used format. When the open operation is successful, a non-NULL descriptor will be returned. Descriptor is the identifier for the subsequent function calls of the 2D GE library.

In contrast to the open function, the close function, “Think2dge_Lib_Close ()”, will be called when users want to close a draw operation. When the close function is called, all resources allocated by the open function will be released.

2.1.3 Target Surface Setting

Before issuing a draw command or blitter, users should set the target surface mode by: Think2d_Target_Set_Mode().

Think2d_Target_Set_Mode() includes setting Target Surface Mode in 0x00, Target Blend Mode in 0x04, Target Base Address in 0x08, Target Stride in 0x0C, Target Resolution in 0x10, Destination Color Key in 0x14, minimum value of Clip Window in 0x18, and maximum value of Clip Window in 0x1C. Table 2-1 lists the registers for the target surface. As shown in Figure 2-1, it describes the relationship between the target base address, target stride, RGB format, and target address generation.

The Clip Window defines a set of coordinates from (Xmin,Ymin) to (Xmax,Ymax), where drawing is permitted. If the Clip Window functionality is not required, the minimum value can be set to (0,0) and the maximum value is equal to (Xmax,Ymax) that allow drawing to be applied on the entire target surface.

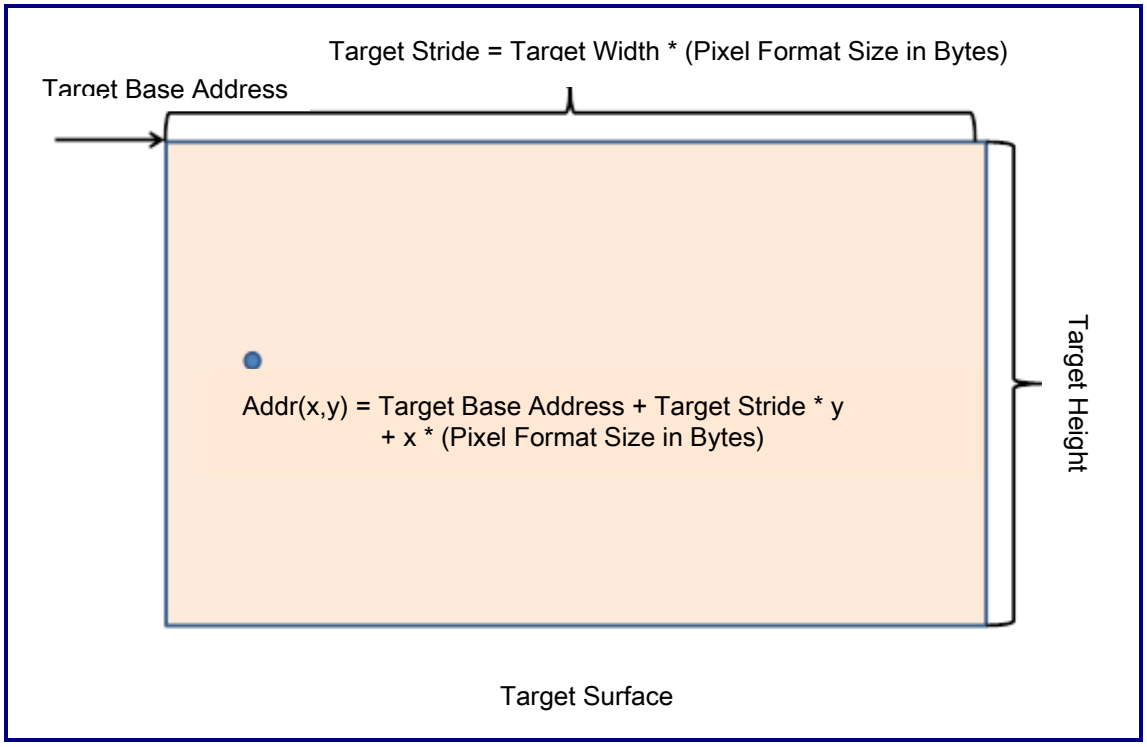

Figure 2-1. Target Surface Layout

2.1.4 Create One Draw Command

Users can set the color for the draw operations. Think2d_Set_Drawing_Color() is used to set the prepared drawing color. Then, users call Think2D_DrawLine() to draw a line, call Think2D_FillRectangle() to fill a rectangle, and call Think2D_DrawRectangle() to draw a rectangle. If all draw parameters are set, users should call Think2D_EmitCommands() to emit.

Users can call Think2d_Set_Drawing_Color() for the preferred color and store to the 0x2C register. Think2D_DrawLine() sets START_XY in 0x24, END_XY in 0x28, and Draw Line bit in the 0x20 register to draw a line. Think2D_FillRectangle() sets START_XY in 0x24, END_XY in 0x28, and fill the Rectangle bit in the 0x20 register to fill a rectangle. Think2D_DrawRectangle() is based on the draw line function to implement drawing a rectangle. Table 2-2 lists the detailed information of the drawing registers.

2.1.5 Create Blitter Command

The blitter command fetches an image from the memory by transforming the source surface to the destination surface. It also performs scaling and rotating of an image, independently on the x and y axes.

Think2D_Blit() sets the blit source address in 0x34, blit source resolution in 0x38, blit destination address in 0x48, and the blit command options, including the rotation, scaling, RGB format, and so on in 0x30 to implement the blitter functionality. Think2D_StretchBlit() sets the blit source address in 0x34, blit source resolution in 0x38, blit destination address in 0x48, destination XY size in 0x4C, scaling Y factor in 0x50, scaling X factor in 0x54, and the blit command options, that include the rotation, scaling, RGB format, and so on in 0x30 to implement the blitter stretching functionality. Then, users should call Think2D_EmitCommands() to emit the commands.

Before calling the Think2D_Blit and Think2D_StretchBlit functions, users have to set the source address and the source stride for the source surface. Table 2-3 lists the detailed information of the blitter register.

2.1.6 Emit Commands

Think2D_EmitCommands() is used to trigger the THINK2D hardware to work correctly. This function will copy the command list to the shared memory pool and then immediately execute these commands. The parameter, b_wait, is suggested setting to '1'. It means that the caller will be blocked until the hardware finishes all commands.

Table 2-1. Target Surface Registers

Offset	Name	Description
0x00	Target Mode	Target surface mode
	Bits	
	31	Assert WM HLOCK in Writes
		0: Transactions in the WM AHB bus are not locked.
		1: Transactions in the WM AHB bus are locked.
		WM burst size in FrameBu_er Writes
	26 - 24	0x00: 16
		0x01: 12
		0x02: 8
		0x03: 4
		0x04: 24
		0x05: 32
		0x06: 48
		0x07: 64
		Assert RM HLOCK in Reads
	23	0: Transactions in the RM AHB bus are not locked.
		1: Transactions in the RM AHB bus are locked.
		RM burst size in FrameBu_er Reads
	18 - 16	0x00: 16
		0x01: 12
		0x02: 8
		0x03: 4
		0x04: 24
		0x05: 32
		0x06: 48
		0x07: 64

Offset	Name	Description
		Enable 16bit ~ 32bit packer
	15	Target Surface
	7 - 0	0x00: RGBX8888 (32bit)
		0x01: RGBA8888
		0x02: XRGB8888
		0x03: ARGB8888
		0x04: RGBA5650 (16bit)
		0x05: RGBA5551
		0x06: RGBA4444
		0x07: Reserved
		0x08: Reserved
		0x09: L8 (Gray Scale)
		0x0A: Reserved
		0x0B: Reserved
		0x0C: Reserved
		0x0D: Reserved
		0x0E: Reserved
		0x0F: Reserved
0x04	TARGET BLEND	Target Blend Mode
	Bits	
	31	Destination Color Key
	30	Colorize Mode
	29 - 20	Reserved
	19 - 16	Destination Blending Mode
		0x0: UNKNOWN
		0x1: ZERO
		0x2: ONE
		0x3: SRCCOLOR
		0x4: INVSRCCOLOR
		0x5: SRCALPHA
		0x6: INVSRCALPHA
		0x7: DESTALPHA
		0x8: INVDESTALPHA
		0x9: DESTCOLOR

Offset	Name	Description
		0xA: INVDESTCOLOR
		0xB: Reserved
		0xC: Reserved
		0xD: Reserved
		0xE: Reserved
		0xF: Reserved
	15 - 0	Source Blending Mode
		0x0: UNKNOWN
		0x1: ZERO
		0x2: ONE
		0x3: SRCCOLOR
		0x4: INVSRCCOLOR
		0x5: SRCALPHA
		0x6: INVSRCALPHA
		0x7: DESTALPHA
		0x8: INVDESTALPHA
		0x9: DESTCOLOR
		0xA: INVDESTCOLOR
		0xB: Reserved
		0xC: Reserved
		0xD: Reserved
		0xE: Reserved
		0xF: Reserved
0x08	TARGET BAR	Target Base Address
	Bits 31 - 0	Base Address of the Drawing Surface (Must be Word Aligned)
0x0C	TARGET STRIDE	Target Stride
	Bits 15 - 0	X Stride (Signed)
0x10	TARGET RESOLXY	Target Resolution
	Bits	
	31 - 16	Resolution Y Size
	15 - 0	Resolution X Size

Offset	Name	Description
0x14	DST COLORKEY	Destination Color Key
	Bits	
	31 - 24	Red value
	23 - 16	Green value
	15 -0	Blue value
0x18	CLIPMIN	Clip Window min. values
	Bits	
	31 - 16	Clipping Y minimum
	15 - 0	Clipping X minimum
0x1C	CLIPMAX	Clip Window max. values
	Bits	
	31 - 16	Clipping Y maximum
	15 - 0	Clipping X maximum

Table 2-2. Drawing Registers

Offset	Name	Description
0x20	DRAW CMD	Draw2D Command
	Bits	
	31	Enable TwinPix (Two pixels per clock cycle)
		0: Draw2D engine draws one pixel per clock cycle.
		1: Draw2D engine draws two pixels per clock cycle (Default).
	2 - 0	Draw Command
		0x00: Draw pixel (Using START XY)
		0x01: Draw line from STARTXY to ENDXY
		0x02: Fill rectangle from STARTXY to ENDXY
0x24	DRAW STARTXY	Drawing Start X,Y
	Bits	
	31 - 16	Start Y
	15 - 0	Start X

Offset	Name	Description
0x28	DRAW ENDXY	End Start X,Y
	Bits	
	31 - 16	END Y
	15 - 0	END X
0x2C	DRAW COLOR	Drawing Color
	Bits	
	31 - 24	Red value
	23 - 16	Green value
	15 - 8	Blue value

Table 2-3. Blitter Registers

Offset	Name	Description
0x30	BLIT CMD	Blitter Command
	Bits	
	31 - 29	Reserved
	28	Force Alpha value
		0: Do not force BLIT FG COLOR alpha value
		1: Force BLIT FG COLOR alpha value
	27	Use Source Color Keying
	26 - 25	Y Scaling mode
		0x00: No Y axis Scaling
		0x01: Up-scaling on Y axis
		0x02: Down-scaling on Y axis
		0x03: Reserved
	24 - 23	X Scaling mode
		0x00: No X axis Scaling
		0x01: Up-scaling on X axis
		0x02: Down-scaling on X axis
		0x03: Reserved
	22 - 20	Rotation (Counter clockwise)
		0x00: 0 degree rotation
		0x01: 90 degree rotation

Offset	Name	Description
		0x02: 180 degree rotation
		0x03: 270 degree rotation
		0x04: mirroring over X axis
		0x05: mirroring over Y axis
	19 - 8	Reserved
	7 - 0	Source Surface
		0x00: RGBX8888 (32bit)
		0x01: RGBA8888
		0x02: XRGB8888
		0x03: ARGB8888
		0x04: RGBA5650 (16bit)
		0x05: RGBA5551
		0x06: RGBA4444
		0x07: Reserved
		0x08: RGBA0008 (A8)
		0x09: L8 (Gray Scale)
		0x0A: Reserved
		0x0B: Reserved
		0x0C: BW1 (A1)
		0x0D: UYVY
		0x0E: Reserved
		0x0F: Reserved
0x34	BLIT SRCADDR	Blitter Source Address
	Bits	
	31 - 0	Start address for the Source Image
0x38	BLIT SRCRESOL	Blitter Source Resolution
	Bits	
	31 - 16	Y size
	15 - 0	X size
0x3C	BLIT SRCOFFSET	Blitter Source Stride
	Bits	
	15 - 0	Signed Stride in bytes

Offset	Name	Description
0x40	BLIT SRC COLORKEY	Blitter Colorkey
	Bits	
	31 - 24	Red value
	23 - 16	Green value
	15 - 8	Blue value
0x44	BLIT FG COLOR	Foreground color
	Bits	
	31 - 24	Red value
	23 - 16	Green value
	15 - 8	Blue value
0x48	BLIT DSTADDR	Destination X,Y
	Bits	
	31 - 16	Destination Y (Signed)
	15 - 0	Destination X (Signed)
0x4C	BLIT DSTYXSIZE	Destination X,Y size
	Bits	
	31 - 16	Destination Y size
	15 - 0	Destination X size
0x50	BLIT SCALE YFN	Scaler Y Function
		Scaling Factor (Fixed point 16.16)
		Upscaling: Source Y size/Destination Y size
		Downscaling: Destination Y size/Source Y size
0x54	BLIT SCALE XFN	Scaler X Function
		X Scaling Factor (Fixed point 16.16)
		Upscaling: Source X size/Destination X size
		Downscaling: Destination X size/Source X size

2.1.7 Examples

Figure 2-2 shows an example of using the library. This example fills a rectangle on the target surface.

```
void main(void)
{
    think2dge_desc_t            *desc;
    think2d_color_t             color;
    think2d_retangle_t          rect;
    int i, j;

    desc = Think2dge_Lib_Open(T2D_RGBA5650);

    if(desc == NULL)
        return;
    /* Set target surface mode base address stride, resolution */
    Think2d_Set_Dest_Surface((think2dge_desc_t*)desc);

    /*Set Clip window*/
    rect.x = 0 ,rect.y = 0;
    rect.w = desc->width , rect.h = desc->height;
    Think2d_Set_Clip_Window(desc,&rect);

    /*Set drawing blend*/
    desc->drawing_flags &=~(THINK2D_DRAW_DST_COLORKEY|THINK2D_DRAW_DSBLEND);
    Think2d_Set_Drawing_Blend(desc);

    /*Set drawing color*/
    color.r = 0xff;
    color.g = 0x00;
    color.b = 0x00;
    Think2d_Set_Drawing_Color((think2dge_desc_t*)desc ,&color);

    printf("think2d Start to test >>>>>>>>>> \n");

    /*FillRectangle*/
    for (i = 0; i < 100; i ++) {
        rect.x = 0 + i*10;
        rect.y = 0 + i *10;
        rect.w = 10;
        rect.h = 10;
        Think2D_FillRectangle((think2dge_desc_t*)desc ,&rect);
    }
    Think2D_EmitCommands((think2dge_desc_t*)desc, 1);
    printf("think2d Test complete. >>>>>>>>>> \n");

    Think2dge_Lib_Close(desc);
}
```

Figure 2-2. Example for Filling Rectangle

Figure 2-3 shows an example of using the library. This example draws a line on the target surface.

```
void main(void)
{
    think2dge_desc_t            *desc;
    think2d_color_t             color;
    think2d_retangle_t          rect;
    think2d_region_t            line;
    int i, j;

    desc = Think2dge_Lib_Open(T2D_RGBA5650);
    if(desc == NULL)
        return;
    /* Set target surface mode base address stride, resolution */
    Think2d_Set_Dest_Surface((think2dge_desc_t*)desc);

    /*Set Clip window*/
    rect.x = 0 ,rect.y = 0;
    rect.w = desc->width , rect.h = desc->height;
    Think2d_Set_Clip_Window(desc,&rect);

    /*Set drawing blend*/
    desc->drawing_flags &=~(THINK2D_DRAW_DST_COLORKEY|THINK2D_DRAW_DSBLEND);
    Think2d_Set_Drawing_Blend(desc);

    printf("think2d Start to test >>>>>>>>>> \n");
    /*draw line*/
    color.r = 0x00 , color.g = 0xff , color.b = 0x00;
    Think2d_Set_Drawing_Color((think2dge_desc_t*)desc ,&color);

    line.x1 = 0 , line.y1 = 0;
    line.y2 = 1080 , line.x2 = 1920;
    Think2D_DrawLine((think2dge_desc_t*)desc ,&line);

    color.r = 0x00 , color.g = 0x00 , color.b = 0xff;
    Think2d_Set_Drawing_Color((think2dge_desc_t*)desc ,&color);

    line.x1 = 1920 , line.y1 = 0;
    line.y2 = 1080 , line.x2 = 0;
    Think2D_DrawLine((think2dge_desc_t*)desc ,&line);

    Think2D_EmitCommands((think2dge_desc_t*)desc, 1);
    printf("think2d Test complete. >>>>>>>>>> \n");

    Think2dge_Lib_Close(desc);
}
```

Figure 2-3. Example for Drawing Line

Figure 2-4 shows an example of using the library. This example transforms an image from the source surface to the target surface.

```
void main(void)
{
    think2dge_desc_t            *desc;
    think2d_color_t             color;
    think2d_retangle_t          rect;
    int                         i,j;
    char                        file_name[20];

    desc = Think2dge_Lib_Open(T2D_RGBA5650);
    if(desc == NULL)
        return;
    /* Set target surface mode base address stride, resolution */
    Think2d_Set_Dest_Surface((think2dge_desc_t*)desc);
    /*Set Clip window*/
    rect.x = 0 ,rect.y = 0;
    rect.w = desc->width , rect.h = desc->height;
    Think2d_Set_Clip_Window(desc,&rect);
    /*Set source blit blend mode*/
    desc->src_blt.blt_src_flags = THINK2D_BLIT_COLORIZE;
    Think2d_Set_Blit_Blend((think2dge_desc_t*)desc);
    /*Create source surface and open image file         */
    /*File format include 8bytes WxH and RGB565 RAW data*/
    sprintf(file_name,"%s","test.bin");
    Think2d_Open_Src_Surface((think2dge_desc_t*)desc,file_name);
    /*Set source stride*/
    desc->src_blt.blt_src_stride = desc->src_blt.blt_src_width * T2D_modesize(desc->bpp_type);
    Think2d_Set_Src_Stride((think2dge_desc_t*)desc);
    /*Set FG color*/
    color.r =0x0,color.g =0xff,color.b=0x0;
    Think2d_Set_FG_Color(desc,&color);
    /*Set rotation*/
    desc->src_blt.blt_src_rotation = T2D_DEG000;
    for(j = 0 ; j < desc->height ;j+=desc->src_blt.blt_src_height){
        for(i = 0 ; i < desc->width ; i+=desc->src_blt.blt_src_width){
            rect.x = 0;
            rect.y = 0;
            rect.w = desc->src_blt.blt_src_width;
            rect.h = desc->src_blt.blt_src_height;
            Think2D_Blit((think2dge_desc_t*)desc ,&rect,i,j);
        }
    }
    Think2D_EmitCommands((think2dge_desc_t*)desc, 1);
    Think2d_Close_Src_Surface((think2dge_desc_t*)desc);
    printf("think2d Test complete. >>>>>>>>>> \n");
    Think2dge_Lib_Close(desc);
    return;
}
```

Figure 2-4. Example for Blitter

Figure 2-5 shows an example of using the library. This example transforms and stretches an image from the source surface to the target surface.

```
void main(void)
{
    think2dge_desc_t            *desc;
    think2d_color_t             color;
    think2d_retangle_t          srect ,drect;
    char                        file_name[20];

    desc = Think2dge_Lib_Open(T2D_RGBA5650);
    if(desc == NULL)
        return;
    /* Set target surface mode base address stride, resolution */
    Think2d_Set_Dest_Surface((think2dge_desc_t*)desc);
    /*Set Clip window*/
    srect.x = 0 ,srect.y = 0;
    srect.w = desc->width , srect.h = desc->height;
    Think2d_Set_Clip_Window(desc,&srect);
    /*Set source blit blend mode*/
    desc->src_blt.blt_src_flags = 0;
    Think2d_Set_Blit_Blend((think2dge_desc_t*)desc);
    /*Create source surface and open image file          */
    /*File format include 8bytes WxH and RGB565 RAW data*/
    sprintf(file_name,"%s","test.bin");
    Think2d_Open_Src_Surface((think2dge_desc_t*)desc,file_name);
    /*Set source stride*/
    desc->src_blt.blt_src_stride = desc->src_blt.blt_src_width * T2D_modesize(desc->bpp_type);
    Think2d_Set_Src_Stride((think2dge_desc_t*)desc);

    /*upscaling*/
    srect.x = 0 , srect.y = 0;
    srect.w = desc->src_blt.blt_src_width , srect.h = desc->src_blt.blt_src_height;

    drect.x = 0 , drect.y = 0;
    drect.w = desc->width , drect.h = desc->height;
    Think2D_StretchBlit(desc ,&srect ,&drect);

    Think2D_EmitCommands((think2dge_desc_t*)desc, 1);
    Think2d_Close_Src_Surface((think2dge_desc_t*)desc);

    printf("think2d Test complete. >>>>>>>>>> \n");
    Think2dge_Lib_Close(desc);
    return;
}
```

Figure 2-5. Example for Stretching Image

Chapter 3

Interface

This chapter contains the following sections:

3.1 Source Code Location

Directory and File	Purpose
arm-linux-3.3\module\include\think2d\think2d_if.h	The header file used in the THINK2DGE driver
arm-linux-3.3\module\think2d\think2d_driver.c	The main driver in THINK2DGE, including interrupt
arm-linux-3.3\module\think2d\gm8139.c	The platform porting file
arm-linux-3.3\module\think2d\think2d_platform.h	The platform header file

Directory and File	Purpose
arm-linux-3.3\module\think2d\lib\think2dge_lib.h	The header file is included by the user application.
arm-linux-3.3\module\ft2dge\lib\think2dge_lib.c	The library is compiled with the user application.
arm-linux-3.3\module\ft2dge\lib\think2d_gfx.c	The library is compiled with DirectFB.
arm-linux-3.3\module\ft2dge\lib\think2d_gfx.h	The header file is included by DirectFB.

The source code of the 2D GE library can be exposed to the user application. Grain Media will release the source code as the open source.

3.2 User Space Interfaces

3.2.1 Error Code

```
0 means success
-1 or NULL means fail
```

3.2.2 Data Structure

The think2dge_desc_t data structure stores the think2d driver file descriptor, target frame buffer file descriptor, target surface resolution, drawing blend mode, destination and source blend mode, frame buffer physical address, frame buffer virtual address, source surface structure, RGB mode, user mode command list, shared memory, and I/O mapping memory from the THINK2DGE driver.

```
typedef struct {
    int fd;                             /*open /dev/think2d fd"*/
    int lcd_fd;                         /*open /dev/fb1 fd"*/
    int width;                          /*target framebuff width*/
    int height;                         /*target framebuff height*/
    unsigned int drawing_flags;         /*drawing mode flag*/
    THINK2D_BLEND_MODE_T v_dstBlend;    /*destination blend mode*/
    THINK2D_BLEND_MODE_T v_srcBlend;    /*source blend mode*/
    unsigned int fb_paddr;              /* frame buffer physical address */
    unsigned int fb_vaddr;              /* frame buffer virtual address */
    think2d_blt_sour_t src_blt;         /*Source surface struct*/
    THINK2D_MODE_T bpp_type;            /*RGB565,RGB888,or ....*/
    think2dge_llst_t *llst;             /*local allocate memory for command list*/
    void *sharemem;                     /*this memory comes from kernel through mapping*/
    void *io_mem;                       /*think2dge io register mapping*/
    unsigned int mapped_sz;             /*sharemem mapping size*/
    unsigned int iomem_sz;              /*io_mem mapping size*/
    pthread_mutex_t  mutex;
} think2dge_desc_t;
```

Figure 3-1. think2dge_desc_t Data Structure

3.2.3 Function Calls

Function Name Summery
think2dge_desc_t *Think2dge_Lib_Open(THINK2D_MODE_T rgb_type);
void Think2dge_Lib_Close(void *descriptor);
int Think2d_Set_Dest_Surface(think2dge_desc_t *tdrv)
int Think2d_Set_Clip_Window(think2dge_desc_t *tdrv,think2d_retangle_t *rect)
int Think2d_Set_Drawing_Blend(think2dge_desc_t *tdrv)
int Think2d_Set_Blit_Blend(think2dge_desc_t *tdrv)
void Think2d_Set_FG_Color(think2dge_desc_t *tdrv,think2d_color_t* color)
int Think2d_Open_Src_Surface(think2dge_desc_t *tdrv , char * file_name)
void Think2d_Close_Src_Surface(think2dge_desc_t *tdrv)
void Think2d_Set_Src_Stride(think2dge_desc_t *tdrv)

Function Name Summery
int Think2d_Target_Set_Mode(think2dge_desc_t *tdrv,struct t2d_target_surface_t *tdata);
int Think2d_Set_Drawing_Color(think2dge_desc_t *tdrv,think2d_color_t* color);
int Think2D_FillRectangle(think2dge_desc_t *tdrv , think2d_retangle_t *rect);
int Think2D_DrawLine(think2dge_desc_t *tdrv,think2d_region_t *line);
int Think2D_DrawRectangle(think2dge_desc_t *tdrv , think2d_retangle_t *rect);
int Think2D_Blit(think2dge_desc_t *tdrv,think2d_retangle_t* rect,int dx,int dy)
int Think2D_StretchBlit(think2dge_desc_t *tdrv,think2d_retangle_t*srect,think2d_retangle_t* drect)
int Think2D_EmitCommands(think2dge_desc_t *tdrv ,int b_wait)

1. think2dge_desc_t *Think2dge_Lib_Open(THINK2D_MODE_T rgb_type)

- Before using the 2DGE module, this function must be called to notify the driver about the RGB type. The LCD frame buffer should be mapped to the descriptor.

Input parameter	rgb_type	rgb_type, please refer to think2dge_lib.h.
Return	• Non-NULL for successful • NULL for failed	

2. void Think2dge_Lib_Close(void *descriptor)

- Close a descriptor

Input parameter	Descriptor	Descriptor given by Think2dge_Lib_Open()
Return	None	

3. int Think2d_Set_Dest_Surface(think2dge_desc_t *tdrv)

- Configure the destination surface, this function calls Think2d_Target_Set_Mode() to set target mode, target base address, target stride, and target resolution.

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
Return	0: Successful -1: Failed	

4. int Think2d_Set_Clip_Window(think2dge_desc_t *tdrv,think2d_retangle_t *rect)

- Configure the destination clip window, this function calls Think2d_Target_Set_Mode() to set the target clip window.

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	rect	rect will be set clip min. XY and max. XY.
Return	0: Successful -1: Failed	

5. int Think2d_Set_Drawing_Blend(think2dge_desc_t *tdrv)

- Configure the drawing blend mode, this function calls Think2d_Target_Set_Mode() to set the target blend.

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open(). This function uses tdrv drawing_flags to set the blend mode, if THINK2D_DRAW_DSBLEND is set. v_dstBlend and v_srcBlend will be taken to set the target blend mode; otherwise, THINK2D_SRCCOPY_MODE will be set.
Return	0: Successful -1: Failed	

6. int Think2d_Set_Blit_Blend(think2dge_desc_t *tdrv)

- Configure the blitter blend mode, this function call Think2d_Target_Set_Mode() to set the target blend.

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open(). This function uses blt_src_flags to set the blend mode, if THINK2D_BLIT_BLEND_ALPHACHANNEL or THINK2D_BLIT_BLEND_COLORALPHA is set. v_dstBlend and v_srcBlend will be taken to set the target blend mode; otherwise, THINK2D_SRCCOPY_MODE will be set.
Return	0: Successful -1: Failed	

7. void Think2d_Set_FG_Color(think2dge_desc_t *tdrv,think2d_color_t* color)

- Configure the foreground color for blitter, before setting, THINK2D_BLIT_COLORIZE must be assigned by Think2d_Set_Blit_Blend().

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	color	Color is in the RGB format.
Return	None	

8. int Think2d_Open_Src_Surface(think2dge_desc_t *tdrv , char * file_name)

- The blitter function needs to set the source surface parameter, such as the source base address and source resolution. User can use this function to create source surface.

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	file_name	It is image raw data file name, the file format must include 8 bytes header (Image width and height) and image raw data.
Return	0: Successful -1: Failed	

9. void Think2d_Close_Src_Surface(think2dge_desc_t *tdrv)

- Close the source surface

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
Return	None	

10. void Think2d_Set_Src_Stride(think2dge_desc_t *tdrv)

- Set the source surface stride

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open(). blt_src_stride is used to set the source stride.
Return	None	

11. int Think2d_Target_Set_Mode(think2dge_desc_t *tdrv,struct t2d_target_surface_t *tdata)

- Set target surface mode

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	tdata	t2d_target_surface_t in think2dge_lib.h
Return	0: Successful -1: Failed	

12. int Think2d_Set_Drawing_Color(think2dge_desc_t *tdrv,think2d_color_t* color)

- Set the color for drawing a line and filling a rectangle

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	color	Color is in the RGB format.
Return	0: Successful -1: Failed	

13. int Think2D_FillRectangle(think2dge_desc_t *tdrv , think2d_retangle_t *rect)

- Set the filling rectangle command

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	rect	Filling rectangle from start XY to end XY
Return	0: Successful -1: Fail	

14. int Think2D_DrawLine(think2dge_desc_t *tdrv,think2d_region_t *line)

- Set the drawing line

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	line	Drawing line from start XY to end XY
Return	0: Successful -1: Failed	

15. int Think2D_DrawRectangle(think2dge_desc_t *tdrv , think2d_retangle_t *rect)

- Set drawing rectangle

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	rect	Drawing rectangle from start XY to end XY
Return	0: Successful -1: Failed	

16. int Think2D_Blit(think2dge_desc_t *tdrv,think2d_retangle_t* rect,int dx,int dy)

- Transform an image from the source surface to the target surface, it performs rotation.

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	rect	Source image start XY width and height
	dx	Destination x value
	dy	Destination y value
Return	0: Successful -1: Failed	

17. int Think2D_StretchBlit(think2dge_desc_t *tdrv,think2d_retangle_t*srect,think2d_retangle_t* drect)

- Stretch an image from the source surface to the target surface

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	srect	Source start XY width and height
	drect	Destination start XY width and height
Return	0: Successful -1: Failed	

18. int Think2D_EmitCommands(think2dge_desc_t *tdrv ,int b_wait)

Input parameters	tdrv	tdrv is given by Think2dge_Lib_Open().
	b_wait	1: Wait for hardware to finish commands 0: Others
Return	0: Successful -1: Failed	

Chapter 4

Unification API

This chapter contains the following sections:

4.1 Introduction

This chapter describes the unification APIs on GM8139 2D graphics accelerator engine. Users can use unification APIs for different 2D graphics accelerator engines to reduce the porting effect. The following sections depict on how to use these APIs.

4.2 Source Code Location

Directory and File	Purpose
arm-linux-3.3\module\think2d\unilib\ft2d_gfx.h	The header file defines the uniform data struct, API, and callback functions.
arm-linux-3.3\module\think2d\unilib\ft2driver_think2d.c	This file is the main interface to call GM8139 2D engine driver.
arm-linux-3.3\module\ft2dge\unilib\ft2driver_think2d.h	This header file is only for ft2driver_think2d.c.
arm-linux-3.3\module\ft2dge\unilib\think2d_demo.c	This file shows how to use unification APIs.

The source code of 2D GE uniform library can be exposed to the user applications. Grain Media will release the source code as the open source.

4.3 Function Calls

No matter what kinds of 2D engine is used, users must use driver_init_device() for mapping the 2D driver command list memory of 2D driver, I/O memory of 2D driver, mapping frame buffer of LCD, hooking API to create the device resource, and using driver_close_device() to close the opened device resource.

The following subsections list the THINK2DGE supported common APIs and hooking APIs.

4.3.1 Common API

1. int driver_init_device(FT2D_GfxDevice *device,
   FT2D_GraphicsDeviceFuncs *funcs,
   FT2D_BPP_T rgb_t)

- Before using the 2DGE module, this function must be called to create the resource. The LCD frame buffer should be mapped.

Input parameter	device	Users need to allocate the FT2D_GfxDevice memory and pass point to it.
	funcs	Users need to allocate the FT2D_GraphicsDeviceFuncs memory and this API will return supported APIs.
	rgb_t	rgb_type, please refer to FT2D_BPP_T in ft2d_gfx.h.
Return	• 0: Successful • -1: Failed	

2. void driver_close_device(FT2D_GfxDevice *device)

- Close the opened device resource

Input parameter	device	FT2D_GfxDevice described opening resource then close it.
Return	None	

4.3.2 Hooking API

Function Name Summery
void (*EngineReset)(void *device);
int (*EmitCommands) (void *device);
int (*CheckState)(void *device);
int (*SetState) (void *device);
int (*FillRectangle) (void *device,FT2DRectangle *rect)
int (*DrawRectangle) (void *device,FT2DRectangle *rect)
int (*DrawLine) (void *device,FT2DRegion *line)
int (*Blit) (void *device , FT2DRectangle *rect, int dx, int dy)
int (*StretchBlit) (void *device , FT2DRectangle *srect, FT2DRectangle *drect)

1. void EngineReset (void *device)

- Reset 2D engine (Not supported)

Input parameter	Device	FT2D_GfxDevice described opening resource.
Return	None	

2. int EmitCommands (void *device)

- After using the drawing or blitting function, it should emit command to execute.

Input parameter	Device	FT2D_GfxDevice described opening resource.
Return	• 0: Successful • -1: Failed	

3. int CheckState (void *device)

- After setting the parameters for drawing or blitting function, such as setting color, it should call this function to check if the 2D driver is supported or not.

Input parameter	device	FT2D_GfxDevice described opening resource.
Return	• 0: Successful • -1: Failed	

4. int SetState (void *device)

- After setting the parameters for drawing or blitting function, such as setting color, it should call this function.

Input parameter	device	FT2D_GfxDevice described opening resource.
Return	• 0: Successful • -1: Failed	

5. int FillRectangle (void *device,FT2DRectangle *rect)

- Fill rectangle with color

Input parameter	device	FT2D_GfxDevice described opening resource.
	rect	Depicted rectangle starting X, Y, width, and height.
Return	• 0: Successful • -1: Failed	

6. int DrawRectangle (void *device,FT2DRectangle *rect)

- Draw rectangle with rectangle size

Input parameter	device	FT2D_GfxDevice described opening resource.
	rect	Depicted rectangle starting X, Y, width, and height.
Return	• 0: Successful • -1: Failed	

7. int DrawLine (void *device,FT2DRegion *line)

- Draw line with start X, Y and end X, Y

Input parameter	device	FT2D_GfxDevice described opening resource.
	line	Depicted start X, Y and end X, Y.
Return	• 0: Successful • -1: Failed	

8. int Blit (void *device , FT2DRectangle *rect, int dx, int dy)

- Blit source surface to target surface

Input parameter	device	FT2D_GfxDevice described opening resource.
	rect	The source surface rectangle is the source which wants to blit to the target surface. The rectangle includes starting X, Y, width, and height.
Return	• 0: Successful • -1: Failed	

9. int StretchBlit (void *device , FT2DRectangle *srect, FT2DRectangle *drect)

- Stretched source surface to target surface.

Input parameter	device	FT2D_GfxDevice described opening resource.
	srect	The source surface rectangle is the source which wants to stretch to the target surface. The rectangle includes starting X, Y, width, and height.
	drect	The target surface rectangle is the target that the source surface wants to stretch. The rectangle includes starting X, Y, width, and height.
Return	• 0: Successful • -1: Failed	

4.3.3 Parameter Setting for Drawing and Blitting Function

The following table lists the setting parameters for the drawing and blitting functions. Please refer to ft2d_gfx.h for the detailed information.

Function	Setting Parameter Flag	Setting Value
FillRectangle	FT2D_COLOR_SET	FT2D_GFX_Setting_Data. color
DrawRectangle	FT2D_DSTINATION_SET	FT2D_GfxDrv_Data. target_sur_data
DrawLine	FT2D_CLIP_SET	FT2D_GFX_Setting_Data.clip
	FT2D_DST_COLORKEY_SET	FT2D_GFX_Setting_Data.dst_color
	FT2D_COLORBLEND_SET	FT2D_GFX_Setting_Data. src_blend
		FT2D_GFX_Setting_Data. dst_blend
Blit	FT2D_DSTINATION_SET	FT2D_GfxDrv_Data. target_sur_data
StretchBlit	FT2D_COLOR_SET	FT2D_GFX_Setting_Data. color
	FT2D_SOURCE_SET	FT2D_GfxDrv_Data. source_sur_data
	FT2D_COLORBLEND_SET	FT2D_GFX_Setting_Data. src_blend
		FT2D_GFX_Setting_Data. dst_blend
	FT2D_DST_COLORKEY_SET	FT2D_GFX_Setting_Data.dst_color
	FT2D_SRC_COLORKEY_SET	FT2D_GFX_Setting_Data.src_color
	FT2D_CLIP_SET	FT2D_GFX_Setting_Data.clip
	FT2D_FG_COLORIZE_SET	Needs to set FT2D_COLOR_SET
	FT2D_FORCE_FG_ALPHA	Needs to set FT2D_COLOR_SET
	FT2D_DEG090_SET	-
	FT2D_DEG180_SET	-
	FT2D_DEG270_SET	-
	FT2D_FLIP_HORIZONTAL_SET	-
	FT2D_FLIP_VERTICAL_SET	-

4.4 Examples

think2dge_demo.c shows the examples that include the drawing line rectangle, filling rectangle, blitting, and stretch-blitting.

```
think2d Demo version  1-0

Usage: think2d_demo [options]

Options:

  --fillrects  fill rectangle
  --drawline   draw line
  --drawrect   draw rectangle
  --blitter    <filename>  blitter filename,file format is WxH header and RAW data(RAW565).
  --blit-stretch <filename> blitter stretch,file format is WxH header and RAW data(RAW565).
  --blitter_90   <filename> blitter rotation 90,file format is WxH header and RAW data(RAW565).
  --blitter_180  <filename> blitter rotation 180,file format is WxH header and RAW data(RAW565).
  --blitter_270  <filename> blitter rotation 270,file format is WxH header and RAW data(RAW565).
  --blitter_flip_v <filename> blitter flip vertical,file format is WxH header and RAW data(RAW565).
  --blitter_flip_h  <filename> blitter flip horizonal,file format is WxH header and RAW
data(RAW565).
```

Chapter 5

DirectFB Supported

This chapter contains the following sections:

5.1 Introduction

This chapter describes DirectFB supported for Think2d. Users can use it to develop GUI with think2d accelerator engines.

5.2 Source Code Location

Directory and File	Purpose
arm-linux-3.3\user\DirectFB\ DirectFB-1.7.1-GM8287-Think2D.tar.gz	This tar file is DirectFB 1.7.1 source code with THINK2D supported.
arm-linux-3.3\user\DirectFB\ Test2D.tar.gz	This file includes Test2D.c , Makefile , directfbrc and *.jpg. Test2D.c is the sample code that can be used to test the performance of think2d. directfbrc is the direct configuration file.

The source code is for DIRECTFB 1.7.1. When users untar DirectFB-1.7.1-GM8287-Think2D.tar.gz , users can find build.sh located in the DirectFB-1.7.1 folder. Users can refer to build.sh to set up the compiler environment.

```
klcheng@ubuntu-1204:~/third_party/DirectFB-1.7.1$ ls
320x240_01.jpg  configure.in            fb.modes      NEWS
aclocal.m4      COPYING                 gfxdrivers    patches
AUTHORS         data                    include       proxy
autogen.sh      depcomp                 inputdrivers  README
build-android   directfb-config         INSTALL       rules
build.sh        directfb-config.in      install-sh    src
ChangeLog       directfb-internal.pc    interfaces    stamp-h1
compile         directfb-internal.pc.in lib           systems
config.guess    directfb.pc             libtool       tests
config.h        directfb.pc.in          ltmain.sh     TODO
config.h.in     directfbrc              m4            tools
config.log      directfb.spec           Makefile      wm
config.status   directfb.spec.in        Makefile.am
config.sub      docs                    Makefile.in
configure       examples                missing
klcheng@ubuntu-1204:~/third_party/DirectFB-1.7.1$ ^C
klcheng@ubuntu-1204:~/third_party/DirectFB-1.7.1$
```