Freescale Semiconductor
User Guide

Document Number: IMX6G2DAPIUG
Rev. L3.0.35_4.1.0, 09/2013

i.MX 6 G2D API User Guide

1. Overview

G2D API (Application Programming Interface) is designed for the purposes of easy to understand and use 2D BLT function. It allows the user to implement the customized applications with the simple interfaces. It is hardware and platform independent for i.MX 6 2D Graphics.
G2D API supports the following features but not limited:

- Simple BLT operation from source to destination
- Alpha blend for source and destination with Porter-Duff rules
- High performance memory copy from source to destination
- Up-scaling and down-scaling from source to destination
- 90/180/270 degree rotation from source to destination
- Horizontal and vertical flip from source to destination
- Enhanced visual quality with dither for pixel precision-loss
- High performance memory clear for destination
- Pixel-level cropping for source surface
- Global alpha blend for source only
- Asynchronous mode and sync
- Contiguous memory allocator
- Support VG engine

G2D API document include the detailed interface description, and sample code for reference.
The API is designed with C-Style and can be used in both C and C++ application.

Contents

2. Enumerations and Structures

This chapter describes all Enumeration and Structure definition in G2D.

2.1 g2d_format Enumeration

The Enumeration describes the pixel format for source and destination

Name	Numeric	Description
G2D_RGB565	0	RGB565 pixel format
G2D_RGBA8888	1	32bit-RGBA pixel format
G2D_RGBX8888	2	32bit-RGBX without alpha
G2D_BGRA8888	3	32bit-BGRA pixel format
G2D_BGRX8888	4	32bit-BGRX without alpha
G2D_NV12	20	Y plane followed by interleaved U/V plane
G2D_I420	21	Y, U, V are within separate planes
G2D_YV12	22	Y, V, U are within separate planes
G2D_NV21	23	Y plane followed by interleaved V/U plane
G2D_YUYV	24	interleaved Y/U/Y/V plane
G2D_YVYU	25	interleaved Y/V/Y/U plane
G2D_UYVY	26	interleaved U/Y/V/Y plane
G2D_VYUY	27	interleaved V/Y/U/Y plane
G2D_NV16	28	Y plane followed by interleaved U/V plane
G2D_NV61	29	Y plane followed by interleaved V/U plane

2.2 g2d_blend_func Enumeration

The Enumeration describes the blend factor for source and destination

Name	Numeric	Description
G2D_ZERO	0	Blend factor with 0
G2D_ONE	1	Blend factor with 1
G2D_SRC_ALPHA	2	Blend factor with source alpha
G2D_ONE_MINUS_SRC_ALPHA	3	Blend factor with 1 - source alpha
G2D_DST_ALPHA	4	Blend factor with destination alpha
G2D_ONE_MINUS_DST_ALPHA	5	Blend factor with 1 - destination alpha

2.3 g2d_cap_mode Enumeration

The Enumeration describes the alternative capability in 2D BLT

Name	Numeric	Description
G2D_BLEND	0	Enable alpha blend in 2D BLT
G2D_DITHER	1	Enable dither in 2D BLT
G2D_GLOBAL_ALPHA	2	Enable global alpha in blend

Note: G2D_GLOBAL_ALPHA is only valid when G2D_BLEND is enabled.

2.4 g2d_rotation Enumeration

The Enumeration describes the rotation mode in 2D BLT

Name	Numeric	Description
G2D_ROTATION_0	0	No rotation
G2D_ROTATION_90	1	Rotation with 90 degree
G2D_ROTATION_180	2	Rotation with 180 degree
G2D_ROTATION_270	3	Rotation with 270 degree
G2D_FLIP_H	4	Horizontal flip
G2D_FLIP_V	5	Vertical flip

2.5 g2d_cache_mode Enumeration

The Enumeration describes the cache operation mode

Name	Numeric	Description
G2D_CACHE_CLEAN	0	Clean the cacheable buffer
G2D_CACHE_FLUSH	1	Clean and invalidate cacheable buffer
G2D_GLOBAL_INVALIDATE	2	Invalidate the cacheable buffer

2.6 g2d_hardware_type Enumeration

The Enumeration describes the supported hardware type

Name	Numeric	Description
G2D_HARDWARE_2D	0	2D hardware type by default
G2D_HARDWARE_VG	1	VG hardware type

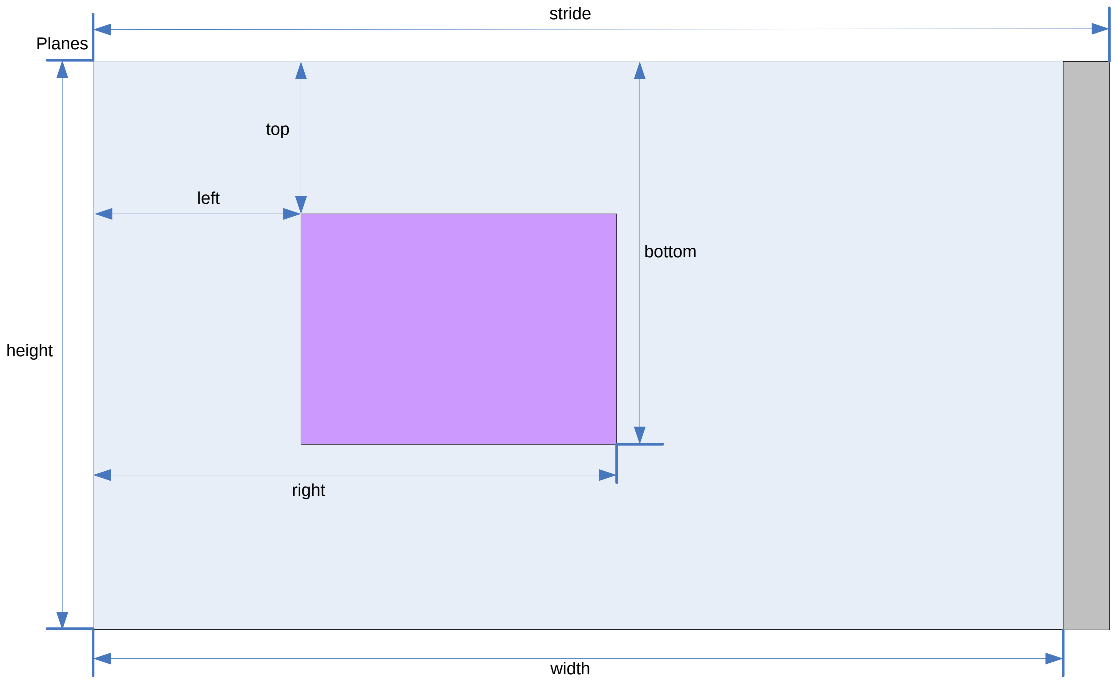
2.7 g2d_surface Structure

The Structure describes the surface with operation attributes

g2d_surface Members	Type	Description
format	g2d_format	Pixel format of surface buffer
planes[3]	int	Physical addresses of surface buffer
left	int	Left offset in blit rectangle
top	int	Top offset in blit rectangle
right	int	Right offset in blit rectangle
bottom	int	Left offset in blit rectangle
stride	int	RGB/Y stride of surface buffer
width	int	Surface width in pixel unit
height	int	Surface height in pixel unit
blendfunc	g2d_blend_func	Alpha blend mode
global_alpha	int	Global alpha value 0~255
clrcolor	int	Clear color is 32bit RGBA
rot	g2d_rotation	Rotation mode

Notes:

1. RGB and YUV formats can be set in source surface, but only RGB format can be set in destination surface.
2. RGB pixel buffer only uses planes [0], buffer address is with 16bytes alignment,
   NV12: Y in planes [0], UV in planes [1], with 64bytes alignment,
   I420: Y in planes [0], U in planes [1], U in planes [2], with 64 bytes alignment
3. The cropped region in source surface is specified with left, top, right and bottom parameters.
4. RGB stride alignment is 16bytes for source and destination surface,
   NV12 stride alignment is 8bytes for source surface, UV stride = Y stride,
   I420 stride alignment is 8bytes for source surface, U stride=V stride = ½ Y stride.
5. G2D_ROTATION_0/G2D_FLIP_H/G2D_FLIP_V shall be set in source surface, and the clockwise rotation degree shall be set in destination surface.
6. Application should calculate the rotated position and set it for destination surface.
7. The geometry definition of surface structure is described as below:



2.8 g2d_buf Structure

The Structure describes the buffer used as g2d interfaces.

g2d_buf Members	Type	Description
buf_handle	void *	The handle associated with buffer
buf_vaddr	void *	Virtual address of the buffer
buf_paddr	int	Physical address of the buffer
buf_size	int	The actual size of the buffer

3. G2D Function Descriptions

3.1 g2d_open

Description:

Open g2d device and return a handle.

Syntax:

```
int g2d_open(void **handle);
```

Parameters:

handle	Pointer to receive g2d device handle

Returns:

Success with 0, fail with -1

3.2 g2d_close

Description:

Close g2d device with the handle.

Syntax:

```
int g2d_close(void *handle);
```

Parameters:

handle	g2d device handle

Returns:

Success with 0, fail with -1

3.3 g2d_make_current

Description:

Set the specific hardware type for current context, default is G2D_HARDWARE_2D

Syntax:

```
int g2d_make_current(void *handle, enum g2d_hardware_type type);
```

Parameters:

handle	g2d device handle
type	g2d hardware type

Returns:

Success with 0, fail with -1

3.4 g2d_clear

Description:

Clear a specific area

Syntax:

```
int g2d_clear(void *handle, struct g2d_surface *area);
```

Parameters:

handle	g2d device handle
area	the area to be cleared

Returns:

Success with 0, fail with -1

3.5 g2d_blit

Description:

G2d blit from source to destination with alternative operation (Blend, Dither, etc)

Syntax:

```
int g2d_blit(void *handle, struct g2d_surface *src, struct g2d_surface *dst);
```

Parameters:

handle	g2d device handle
src	source surface
dst	destination surface

Returns:

Success with 0, fail with -1

3.6 g2d_copy

Description:

G2d copy with specified size

Syntax:

```
int g2d_copy(void *handle, struct g2d_buf *d, struct g2d_buf* s, int size);
```

Parameters:

handle	g2d device handle
d	destination buffer
s	source buffer
size	copy bytes

Limitations:

If the destination buffer is cacheable, it must be invalidated before g2d_copy due to the alignment limitation of g2d driver.

Returns:

Success with 0, fail with -1

3.7 g2d_query_cap

Description:

Query the alternative capability enablement

Syntax:

int g2d_query_cap(void *handle, enum g2d_cap_mode cap, int *enable);

Parameters:

handle	g2d device handle
cap	g2d capability to query
enable	Pointer to receive g2d capability enablement

Returns: Success with 0, fail with -1

3.8 g2d_enable

Description:

Enable g2d capability with the specific mode

Syntax:

int g2d_enable(void *handle, enum g2d_cap_mode cap);

Parameters:

handle	g2d device handle
cap	g2d capability to enable

Returns:

Success with 0, fail with -1

3.9 g2d_disable

Description:

Enable g2d capability with the specific mode

Syntax:

int g2d_disable (void *handle, enum g2d_cap_mode cap);

Parameters:

handle	g2d device handle
cap	g2d capability to disable

Returns:

Success with 0, fail with -1

3.10 g2d_cache_op

Description:

Perform cache operations for the cacheable buffer allocated through g2d driver

Syntax:

```
int g2d_cache_op (struct g2d_buf *buf, enum g2d_cache_mode op);
```

Parameters:

buf	the buffer to be handled with cache operations
op	cache operation type

Returns:

Success with 0, fail with -1

3.11 g2d_alloc

Description:

Allocate a buffer through g2d device

Syntax:

```
struct g2d_buf *g2d_alloc(int size, int cacheable);
```

Parameters:

size	allocated bytes
cacheable	0, non-cacheable, 1, cacheable attribute defined by system

Returns:

Success with valid g2d buffer pointer, fail with 0

3.12 g2d_free

Description:

Free the buffer through g2d device

Syntax:

```
int g2d_free(struct g2d_buf *buf);
```

Parameters:

buf	g2d buffer to free

Returns:

Success with 0, fail with -1

3.13 g2d_flush

Description:
Flush g2d command and return without completing pipeline.

Syntax:
```
int g2d_flush (void *handle);
```

Parameters:

handle	g2d device handle

Returns:
Success with 0, fail with -1

3.14 g2d_finish

Description:
Flush g2d command and then return when pipeline is finished

Syntax:
```
int g2d_finish (void *handle);
```

Parameters:

handle	g2d device handle

Returns:
Success with 0, fail with -1

4. Sample code for G2D API usage

This chapter gives the brief prototype codes with g2d api

4.1 Color space conversion from YUV to RGB

```
g2d_open(&handle);

src.planes[0] = buf_y;
src.planes[1] = buf_u;
src.planes[2] = buf_v;
src.left = crop.left;
src.top = crop.top;
src.right = crop.right;
src.bottom = crop.bottom;
src.stride = y_stride;
src.width = y_width;
src.height = y_height;
src.rot    = G2D_ROTATION_0;
src.format = G2D_I420;

dst.planes[0] = buf_rgba;
dst.left = 0;
dst.top = 0;
dst.right = disp_width;
dst.bottom = disp_height;
dst.stride = disp_width;
dst.width = disp_width;
dst.height = disp_height;
dst.rot    = G2D_ROTATION_0;
dst.format = G2D_RGBA8888;

g2d_blit(handle, &src, &dst);
g2d_finish(handle);

g2d_close(handle);
```

4.2 Alpha blend in Source Over mode

```
g2d_open(&handle);

src.planes[0] = src_buf;
src.left = 0;
src.top = 0;
src.right = test_width;
src.bottom = test_height;
src.stride = test_width;
src.width = test_width;
src.height = test_height;
src.rot    = G2D_ROTATION_0;
src.format = G2D_RGBA8888;
src.blendfunc = G2D_ONE;

dst.planes[0] = dst_buf;
dst.left = 0;
dst.top = 0;
dst.right = test_width;
dst.bottom = test_height;
dst.stride = test_width;
dst.width = test_width;
dst.height = test_height;
dst.format = G2D_RGBA8888;
dst.rot    = G2D_ROTATION_0;
dst.blendfunc = G2D_ONE_MINUS_SRC_ALPHA;

g2d_enable(handle,G2D_BLEND);
g2d_blit(handle, &src, &dst);
g2d_finish(handle);
g2d_disable(handle,G2D_BLEND);

g2d_close(handle);
```

4.3 Source cropping and destination rotation

```
g2d_open(&handle);

src.planes[0] = src_buf;
src.left = crop.left;
src.top = crop.left;
src.right = crop.right;
src.bottom = crop.bottom;
src.stride = src_stride;
src.width = src_width;
rc.height = src_height;
src.format = G2D_RGBA8888;
src.rot    = G2D_ROTATION_0;//G2D_FLIP_H or G2D_FLIP_V

dst.planes[0] = dst_buf;
dst.left = 0;
dst.top = 0;
dst.right = dst_width;
dst.bottom = dst_height;
dst.stride = dst_width;
dst.width = dst_width;
dst.height = dst_height;
dst.format = G2D_RGBA8888;
dst.rot    = G2D_ROTATION_90;

g2d_blit(handle, &src, &dst);
g2d_finish(handle);

g2d_close(handle);
```

Document Number: IMX6G2DAPIUG
Rev. L3.0.35_4.1.0
09/2013