Freescale Semiconductor

Document Number: i.MX 6 2D API
Rev. 1.2, 05/2013

i.MX 6 2D API

(Hardware and Platform Independent API for i.MX 6 2D Graphics)

Contents

1 Overview

The software includes 2D control mechanisms which allow the user to implement customized applications and drivers using 2D specific Application Programming Interface (API).

The API relies on a Hardware Abstraction Layer (HAL) designed to simplify the complexity of graphics software development and hide platform and architecture dependent code. HAL is portable across different platforms and architectures without changing the core—all platform and architecture dependent code lives within a tiny shell around the core.

Inside HAL's application layer is the API used by the developer. The kernel layer, which controls the hardware, is completely hidden. The Platform Dependent Code contains the platform specific code. This is the code portion you use to port to a new platform. The Hardware Dependent Code contains all the hardware dependent code.

Multiple applications and/or threads within an application can access HAL. Each separate thread, which requires access to HAL, has its own context. Context switching between different threads is completely hidden and is handled by a combination of code in the application layer and kernel layer. Communication between the application and kernel layer is handled through generic device I/O control calls.

Details of 2D API are described in this document.
First, parameter types, return codes and enumerations, and relevant structures are described.
Next, various types of API objects are detailed. These include objects for:

- OS Operating system specific operations and functions
- HAL Hardware specific functions
- 2D 2D operations, such as blit, brush, and alpha blending operations
- SURF Surface manipulation functions
- RECT Rectangle functions

Finally, a descriptive summary of common 2D operations is included for reference.

2 API Components and Conventions

The HAL is written in ANSI C code to provide the greatest platform portability. Opaque structures are used to hide all aspects of the internals. Every object is described by a pointer to a structure whose contents are unknown. Whenever an API entry is called, a pointer to the corresponding object is passed in.

2.1 Naming Conventions

This document uses a naming conventions scheme wherein definitions are preceded by three-letter indicators that start with 'gc,' for graphics core, followed by a suffix letter that represents the nature of the definition. The three-letter indicators are then followed by an upper case definition name. An exception to the upper case rule occurs when listing function names, which receive an initial capitalization only (for example gcoOS_Construct). Where necessary, an underscore (_) is used to separate words.

Suffix Letters	Definition

gce	**Graphics Core Enumerated Types** for types defined with the keyword enum. For example: gceDEPTH_MODE.
gcm	**Graphics Core Macros** for macros defined with the keyword define (but not simple values). For example: gcmTRACE_ZONE
gco	**Graphics Core Objects** for objects defined with the keyword struct. For example: gcoHARDWARE.
gcs	**Graphics Core Structures** for types defined with the keyword struct (but not objects). For example: gcs*UPPER_CASE_NAME*
gct	**Graphics Core Types** for simple types defined with the keyword typedef (but not enums, structs, or unions). For example: gctFIXED_POINT
gcu	**Graphics Core Union Types** for types defined with the keyword union. For example: gcu*UPPER_CASE_NAME*
gcv	**Graphics Core Values** for values defined with the keywords defined or enum. For example: gcvSTATUS_BUFFER_TOO_SMALL

Important things to remember

- Objects differ from regular structures in that they have gcsOBJECT member defined as the very first member.
- gco definitions are always defined as pointers to objects.
- gct, gce, gcu, and gcs are never directly defined as pointers. Add a _PTR postfix at the end of the type name to define a pointer. For example:

```
typedef int gctINT                  // Never a pointer directly
typedef gctINT * gctINT_PTR         // Correct pointer definition
```

2.2 Base Objects

All objects in the library are *based* on **gcoOBJECT** by including a member of **gcoOBJECT** type as the first member. Every object has its own object type defined. The Base Object includes:

```
Base Object
      gceOBJECT_TYPE
```

Most functions in HAL API return a status value of **gceSTATUS** type.

2.3 Common Parameter Types

There are a number of common types. Not all are used by the 2D API.

Name	Typedef	Value
gctBOOL	int	FALSE or TRUE
gctCHAR	char	A 8 bit character value
gcsCOUNT_STRING	structure	A counted string (like Pascal)
gctFIXED_POINT	signed int	A 16.16 fixed point number
gctFLOAT	float	A single precision floating point number
gctINT	int	A signed integer
gctINT8	signed char	A signed 8-bit value
gctINT16	signed short	A signed 16-bit value
gctINT32	signed int	A signed 32-bit value
gctINT64	signed long long	A signed 64-bit value
gctSIZE_T	unsigned long	A size of something
gctUINT	unsigned int	An unsigned integer
gctUINT8	unsigned char	An unsigned 8-bit value
gctUINT16	unsigned short	An unsigned 16-bit value
gctUINT32	unsigned int	An unsigned 32-bit value
gctUINT64	unsigned long long	An unsigned 64-bit value

2.4 Common Parameter Pointer Types

There are a number of common types. Not all are used by the 2D API.

Name	**Typedef**	**Value**
gctBOOL_PTR	gctBOOL *	A pointer to a gctBOOL boolean
gctCONST_POINTER	const void *	A generic pointer to constant data
gctCONST_STRING	const char *	A pointer to a constant zero-terminated string
gctFILE	void *	A pointer t a file
gctFLOAT_PTR	float *	Pointer to a floating point value
gctHANDLE	void *	A handle for the OS
gctINT_PTR	gctINT *	Pointer to a signed integer
gctINT8_PTR	gctINT8 *	Pointer to a signed 8-bit value
gctINT16_PTR	gctINT16 *	Pointer to a signed 16-bit value
gctINT32_PTR	gctINT32 *	Pointer to a signed 32-bit value
gctINT64_PTR	gctINT64 *	Pointer to a signed 64-bit value
gctPHYS_ADDR	void *	A pointer to a physical address
gctPOINTER	void *	A generic pointer
gctSIZE_T_PTR	gctSIZE_T *	Pointer to a variable containing a size of something
gctSTRING	void *	A pointer to string data
gctUINT_PTR	gctUINT *	Pointer to an unsigned integer
gctUINT8_PTR	gctUINT8 *	Pointer to an unsigned 8-bit value
gctUINT16_PTR	gctUINT16 *	Pointer to an unsigned 16-bit value
gctUINT32_PTR	gctUINT32 *	Pointer to an unsigned 32-bit value
gctUINT64_PTR	gctUINT64 *	Pointer to an unsigned 64-bit value

2.5 Return Status Enumeration

2.5.1 gceSTATUS Enumeration Return Status Codes

Most functions in HAL API return a status value of **gceSTATUS** type. All API functions return the status of the command and will report **gcvSTATUS_OK** if successful with no errors. Possible status values include the values in the table below. Not all values may apply to 2D operations.

gceSTATUS String Value	Numeric	Description
gcvSTATUS_OK	0	No error
gcvSTATUS_FALSE	0	
gcvSTATUS_TRUE	1	
gcvSTATUS_NO_MORE_DATA	2	No error; no more data is available for an enumeration function.
gcvSTATUS_CACHED	3	No error; the requested state has been cached.
gcvSTATUS_MIPMAP_TOO_LARGE	4	
gcvSTATUS_NAME_NOT_FOUND	5	
gcvSTATUS_NOT_OUR_INTERRUPT	6	
gcvSTATUS_MISMATCH	7	
gcvSTATUS_MIPMAP_TOO_SMALL	8	
gcvSTATUS_LARGER	9	Item or String 1 is larger than Item / String 2
gcvSTATUS_SMALLER	10	Item or String 1 is smaller than Item / String 2
gcvSTATUS_CHIP_NOT_READY	11	
gcvSTATUS_NEED_CONVERSION	12	
gcvSTATUS_SKIP	13	
gcvSTATUS_DATA_TOO_LARGE	14	
gcvSTATUS_INVALID_CONFIG	15	
gcvSTATUS_CHANGED	16	
gcvSTATUS_NOT_SUPPORT_DITHER	17	
gcvSTATUS_EXECUTED	18	
gcvSTATUS_TERMINATE	19	
gcvSTATUS_INVALID_ARGUMENT	-1	An API entry was called with an invalid argument
gcvSTATUS_INVALID_OBJECT	-2	An API entry was called with an invalid object
gcvSTATUS_OUT_OF_MEMORY	-3	Out of memory
gcvSTATUS_MEMORY_LOCKED	-4	Trying to free locked memory
gcvSTATUS_MEMORY_UNLOCKED	-5	Trying to unlock already unlocked memory
gcvSTATUS_HEAP_CORRUPTED	-6	Fatal heap corruption error
gcvSTATUS_GENERIC_IO	-7	Generic I/O error
gcvSTATUS_INVALID_ADDRESS	-8	An API entry was called with an invalid address

gcvSTATUS_CONTEXT_LOSSED	-9	
gcvSTATUS_TOO_COMPLEX	-10	The operation is too complex for the hardware to handle
gcvSTATUS_BUFFER_TOO_SMALL	-11	The command buffer or command queue overflows
gcvSTATUS_INTERFACE_ERROR	-12	A platform interface returned an error
gcvSTATUS_NOT_SUPPORTED	-13	Operation is not supported
gcvSTATUS_MORE_DATA	-14	An API entry was called with not enough data
gcvSTATUS_TIMEOUT	-15	The process timed out
gcvSTATUS_OUT_OF_RESOURCES	-16	Out of system resources
gcvSTATUS_INVALID_DATA	-17	
gcvSTATUS_INVALID_MIPMAP	-18	
gcvSTATUS_NOT_FOUND	-19	
gcvSTATUS_NOT_ALIGNED	-20	
gcvSTATUS_INVALID_REQUEST	-21	
gcvSTATUS_GPU_NOT_RESPONDING	-22	
gcvSTATUS_TIMER_OVERFLOW	-23	
gcvSTATUS_VERSION_MISMATCH	-24	
gcvSTATUS_LOCKED	-25	
gcvSTATUS_INTERRUPTED	-26	
gcvSTATUS_DEVICE	-27	
gcvSTATUS_NOT_MULTI_PIPE_ALIGNED	-28	

2.6 Enumerations used in API

2.6.1 gce2D_GLOBAL_COLOR_MULTIPLY_MODE Enumeration

Used in objects: gco2D_SetPixelMultiplyModesAdvanced.

gce2D_GLOBAL_COLOR_MULTIPLY_MODE String Values	**Numeric**	**Description**
gcv2D_GLOBAL_COLOR_MULTIPLY_DISABLE	0	
gcv2D_GLOBAL_COLOR_MULTIPLY_ALPHA	1	
gcv2D_GLOBAL_COLOR_MULTIPLY_COLOR	2	

2.6.2 gce2D_PIXEL_COLOR_MULTIPLY_MODE Enumeration

Used in objects: gco2D_SetPixelMultiplyModesAdvanced.

gce2D_PIXEL_COLOR_MULTIPLY_MODE String Values	Numeric	Description
gcv2D_COLOR_MULTIPLY_DISABLE	0	
gcv2D_COLOR_MULITPLY_ENABLE	1	

2.6.3 gce2D_PORTER_DUFF_RULE Enumeration

Alpha Blending Porter Duff Rules. Used in objects: gco2D_SetPorterDuffBlending.

gce2D_PORTER_DUFF_RULE String Values	Numeric	Description
gcvPD_CLEAR	0	
gcvPD_SRC	1	
gcvPD_SRC_OVER	2	
gcvPD_DST_OVER	3	
gcvPD_SRC_IN	4	
gcvPD_DST_IN	5	
gcvPD_SRC_OUT	6	
gcvPD_DST_OUT	7	
gcvPD_SRC_ATOP	8	
gcvPD_DST_ATOP	9	
gcvPD_ADD	10	
gcvPD_XOR	11	
gcvPD_DST	12	

2.6.4 gce2D_QUERY Enumeration

Used in objects: gco2D_QueryU32.

gce2D_QUERY String Values	Numeric	Description
gcv2D_QUERY_RGB_ADDRESS_MAX_ALIGN	0	
gcv2D_QUERY_RGB_STRIDE_MAX_ALIGN	1	
gcv2D_QUERY_YUV_ADDRESS_MAX_ALIGN	2	
gcv2D_QUERY_YUV_STRIDE_MAX_ALIGN	3	

2.6.5 gce2D_STATE Enumeration

Used in objects: gco2D_SetStateU32.

gce2D_STATE String Values	Numeric	Description
gcv2D_STATE_SPECIAL_FILTER_MIRROR_MODE	1	

2.6.6 gce2D_TILE_STATUS_CONFIG Enumeration

Used in objects: gco2D_SetSourceTileStatus.

gce2D_TILE_STATUS_CONFIG String Values	Numeric	Description
gcv2D_TSC_DISABLE	0	
gcv2D_TSC_ENABLE	1	
gcv2D_TSC_COMPRESSED	2	
gcv2D_TSC_DOWN_SAMPLER	4	

2.6.7 gce2D_TRANSPARENCY Enumeration

Valid only with Pixel Engine 2.0 or later. Used in objects: gco2D_SetTransparencyAdvancedEx.

gce2D_TRANSPARENCY String Values	Numeric	Description
gcv2D_OPAQUE	0	
gcv2D_KEYED	1	
gcv2D_MASKED	2	

2.6.8 gce2D_YUV_COLOR_MODE Enumeration

Used in objects: gco2D_SetYUVColorMode.

gce2D_YUV_COLOR_MODE String Values	Numeric	Description
gcv2D_YUV_601	0	
gcv2D_YUV_709	1	Not supported on i.MX 6

2.6.9 gceFEATURE Enumeration

The super set of features listed below includes features never provided in 2D cores.

Used in objects: gcoHAL_IsFeatureAvailable.

gceFEATURE String Values	Numeric	Description
gcvFEATURE_PIPE_2D	0	Hardware supports 2D core.
gcvFEATURE_PIPE_3D	1	*not applicable for 2D cores*
gcvFEATURE_PIPE_VG	2	*not applicable for 2D cores*
gcvFEATURE_DC	3	*not applicable for 2D cores*
gcvFEATURE_HIGH_DYNAMIC_RANGE	4	*not applicable for 2D cores*
gcvFEATURE_MODULE_CG	5	*not applicable for 2D cores*
gcvFEATURE_MIN_AREA	6	*not applicable for 2D cores*
gcvFEATURE_BUFFER_INTERLEAVING	7	*not applicable for 2D cores*
gcvFEATURE_BYTE_WRITE_2D	8	*not applicable for 2D cores*
gcvFEATURE_ENDIANNESS_CONFIG	9	*not applicable for 2D cores*

gcvFEATURE_DUAL_RETURN_BUS	10	*not applicable for 2D cores*
gcvFEATURE_DEBUG_MODE	11	*not applicable for 2D cores*
gcvFEATURE_YUY2_RENDER_TARGET	12	*not applicable for 2D cores*
gcvFEATURE_FRAGMENT_PROCESSOR	13	*not applicable for 2D cores*
gcvFEATURE_2DPE20	14	2D Pixel Engine (PE) 2.0 is available.
gcvFEATURE_FAST_CLEAR	15	*not applicable for 2D cores*
gcvFEATURE_YUV420_TILER	16	*not applicable for 2D cores*
gcvFEATURE_YUY2_AVERAGING	17	*not applicable for 2D cores*
gcvFEATURE_FLIP_Y	18	*not applicable for 2D cores*
gcvFEATURE_EARLY_Z	19	*not applicable for 2D cores*
gcvFEATURE_Z_COMPRESSION	20	*not applicable for 2D cores*
gcvFEATURE_MSAA	21	*not applicable for 2D cores*
gcvFEATURE_SPECIAL_ANTI_ALIASING	22	*not applicable for 2D cores*
gcvFEATURE_SPECIAL_MSAA_LOD	23	*not applicable for 2D cores*
gcvFEATURE_422_TEXTURE_COMPRESSION	24	*not applicable for 2D cores*
gcvFEATURE_DXT_TEXTURE_COMPRESSION	25	*not applicable for 2D cores*
gcvFEATURE_ETC1_TEXTURE_COMPRESSION	26	*not applicable for 2D cores*
gcvFEATURE_CORRECT_TEXTURE_CONVERTER	27	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_8K	28	*not applicable for 2D cores*
gcvFEATURE_SCALER	29	*not applicable for 2D cores*
gcvFEATURE_YUV420_SCALER	30	2D core supports YUV420 scaler.
gcvFEATURE_SHADER_HAS_W	31	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_SIGN	32	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_FLOOR	33	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_CEIL	34	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_SQRT	35	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_TRIG	36	*not applicable for 2D cores*
gcvFEATURE_VAA	37	*not applicable for 2D cores*
gcvFEATURE_HZ	38	*not applicable for 2D cores*
gcvFEATURE_CORRECT_STENCIL	39	*not applicable for 2D cores*
gcvFEATURE_VG20	40	*not applicable for 2D cores*
gcvFEATURE_VG_FILTER	41	*not applicable for 2D cores*
gcvFEATURE_VG21	42	*not applicable for 2D cores*
gcvFEATURE_VG_DOUBLE_BUFFER	43	*not applicable for 2D cores*
gcvFEATURE_MC20	44	*not applicable for 2D cores*
gcvFEATURE_SUPER_TILED	45	*not applicable for 2D cores*
gcvFEATURE_2D_FILTERBLIT_PLUS_ALPHABLEND	46	2D core supports filter blit plus alpha blending.
gcvFEATURE_2D_DITHER	47	2D core supports dithering.
gcvFEATURE_2D_A8_TARGET	48	2D core supports A8 as target format.

gcvFEATURE_2D_FILTERBLIT_FULLROTATION	49	2D core supports filter plus full rotations.
gcvFEATURE_2D_BITBLIT_FULLROTATION	50	2D core supports blit plus full rotations.
gcvFEATURE_WIDE_LINE	51	*not applicable for 2D cores*
gcvFEATURE_FC_FLUSH_STALL	52	*not applicable for 2D cores*
gcvFEATURE_FULL_DIRECTFB	53	2D core supports full DirectFB mode.
gcvFEATURE_HALF_FLOAT_PIPE	54	*not applicable for 2D cores*
gcvFEATURE_LINE_LOOP	55	*not applicable for 2D cores*
gcvFEATURE_2D_YUV_BLIT	56	2D core supports blit plus YUV formats.
gcvFEATURE_2D_TILING	57	2D core supports tiling surface.
gcvFEATURE_NON_POWER_OF_TWO	58	*not applicable for 2D cores*
gcvFEATURE_3D_TEXTURE	59	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_ARRAY	60	*not applicable for 2D cores*
gcvFEATURE_TILE_FILLER	61	*not applicable for 2D cores*
gcvFEATURE_LOGIC_OP	62	*not applicable for 2D cores*
gcvFEATURE_COMPOSITION	63	*not applicable for 2D cores*
gcvFEATURE_MIXED_STREAMS	64	*not applicable for 2D cores*
gcvFEATURE_2D_MULTI_SOURCE_BLT	65	2D core supports multi source blit.
gcvFEATURE_END_EVENT	66	*not applicable for 2D cores*
gcvFEATURE_VERTEX_10_10_10_2	67	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_10_10_10_2	68	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_ANISOTROPIC_FILTERING	69	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_FLOAT_HALF_FLOAT	70	*not applicable for 2D cores*
gcvFEATURE_2D_ROTATION_STALL_FIX	71	*not applicable for 2D cores*
gcvFEATURE_2D_MULTI_SOURCE_BLT_EX	72	2D core supports the externsion of multi source blit.
gcvFEATURE_BUG_FIXES10	73	*not applicable for 2D cores*
gcvFEATURE_2D_MINOR_TILING	74	2D core supports minor tiling surface.
gcvFEATURE_TEX_COMPRRESSION_SUPERTILED	75	*not applicable for 2D cores*
gcvFEATURE_FAST_MSAA	76	*not applicable for 2D cores*
gcvFEATURE_BUG_FIXED_INDEXED_TRIANGLE_STRIP	77	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_TILED_READ	78	*not applicable for 2D cores*
gcvFEATURE_DEPTH_BIAS_FIX	79	*not applicable for 2D cores*
gcvFEATURE_RECT_PRIMITIVE	80	*not applicable for 2D cores*
gcvFEATURE_BUG_FIXES11	81	*not applicable for 2D cores*
gcvFEATURE_SUPERTILED_TEXTURE	82	*not applicable for 2D cores*
gcvFEATURE_2D_NO_COLORBRUSH_INDEX8	83	2D core dose not support color brush and index8 format.
gcvFEATURE_RS_YUV_TARGET	84	*not applicable for 2D cores*
gcvFEATURE_2D_FC_SOURCE	85	2D core supports source surface with fast clear status.

gcvFEATURE_PE_DITHER_FIX	86	*not applicable for 2D cores*
gcvFEATURE_2D_YUV_SEPARATE_STRIDE	87	2D core supports seprate strides for planar YUV formats.
gcvFEATURE_FRUSTUM_CLIP_FIX	88	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_LINEAR	89	*not applicable for 2D cores*
gcvFEATURE_TEXTURE_YUV_ASSEMBLER	90	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_ATOMIC	91	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_INSTRUCTION_CACHE	92	*not applicable for 2D cores*
gcvFEATURE_SHADER_ENHANCEMENTS2	93	*not applicable for 2D cores*
gcvFEATURE_BUG_FIXES7	94	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_RTNE	95	*not applicable for 2D cores*
gcvFEATURE_SHADER_HAS_EXTRA_INSTRUCTIONS2	96	*not applicable for 2D cores*

2.6.10 gceFILE_MODE Enumeration Return Status Codes

Used in objects: gcoOS_Open.

gceFILE_MODE String Values	Numeric	Description
gcvFILE_CREATE	0	Create a new file; will overwrite an existing file of the same name.
gcvFILE_APPEND	1	Append to an existing file; will create a new file if none exists.
gcvFILE_READ	2	Open an existing file for read only.
gcvFILE_APPENDTEXT	3	Append to an exisiting text file or create a new text file if there is no exisiting file.
gcvFILE_CREATETEXT	4	Create a new text file; will overwrite an existing text file of the same name.
gcvFILE_READTEXT	5	Open an existing text file for read only.

2.6.11 gceFILTER_PASS_TYPE Enumeration

Used in objects: gco2D_SetUserFilterKernel.

gceFILTER_PASS_TYPE String Values	Numeric	Description
gcvFILTER_HOR_PASS	0	
gcvFILTER_VER_PASS	1	

2.6.12 gceFILTER_TYPE Enumeration

Used in objects: gco2D_SetFilterType.

gceFILTER_PASS_TYPE String Values	Numeric	Description
gcvFILTER_SYNC	0	

gcvFILTER_BLUR	1	
gcvFILTER_USER	2	

2.6.13 gceHARDWARE_TYPE Enumeration

Used in objects: gcoHAL_GetHardwareType, gcoHAL_SetHardwareType.

gceHARDWARE_TYPE String Values	Numeric	Description
gcvHARDWARE_INVALID	0	
gcvHARDWARE_3D	1	
gcvHARDWARE_2D	2	
gcvHARDWARE_VG	4	
gcvHARDWARE_3D2D		= gcvHARDWARE_3D | gcvHARDWARE_2D

2.6.14 gcePOOL Enumeration

The pool, from which, you want your new surface to allocate. Used in objects: gcoSURF_Contstruct.

gcePOOL String Values	Numeric	Description
gcvPOOL_UNKNOWN	0	
gcvPOOL_DEFAULT	1	
gcvPOOL_LOCAL	2	
gcvPOOL_LOCAL_INTERNAL	3	
gcvPOOL_LOCAL_EXTERNAL	4	
gcvPOOL_UNIFIED	5	
gcvPOOL_SYSTEM	6	
gcvPOOL_VIRTUAL	7	
gcvPOOL_USER	8	
gcvPOOL_CONTIGUOUS	9	
gcvPOOL_DEFAULT_FORCE_CONTIGUOUS	10	
gcvPOOL_DEFAULT_FORCE_CONTIGUOUS_ CACHEABLE	11	
gcvPOOL_NUMBER_OF_POOLS	12	

2.6.15 gceSURF_BLEND_FACTOR_MODE Enumeration

Used in objects: gco2D_EnableAlphaBlend, gco2D_EnableAlphaBlendAdvanced, gcoSURF_EnableAlphaBlend.

gceSURF_BLEND_FACTOR_MODE String Values	Numeric	Description
		Porter Duff Blending modes
gcvBLEND_CLEAR	0	Fsrc 0 Fdst 0

gcvBLEND_SRC	1	Fsrc 1 Fdst 0
gcvBLEND_DST	2	Fsrc 0 Fdst 1
gcvBLEND_SRC_OVER_DST	3	Fsrc 1 Fdst 1 – Asrc
gcvBLEND_DST_OVER_SRC	4	Fsrc 1 – Adst Fdst 1
gcvBLEND_SRC_IN_DST	5	Adst Fdst 0
gcvBLEND_DST_IN_SRC	6	Fsrc 0 Asrc
gcvBLEND_SRC_OUT_DST	7	Fsrc 1 – Adst Fdst 0
gcvBLEND_DST_OUT_SRC	8	Fsrc 0 Fdst 1 – Asrc
gcvBLEND_SRC_ATOP_DST	9	Adst Fdst 1 – Asrc
gcvBLEND_DST_ATOP_SRC	10	Fsrc 1 – Adst Asrc
gcvBLEND_SRC_XOR_DST	11	Fsrc 1 – Adst Fdst 1 – Asrc
		Special blending modes
gcvBLEND_SET	12	DST = 1
gcvBLEND_SUB	13	DST = DST * (1 – SRC)

2.6.16 gceSURF_FORMAT Enumeration

Used in objects: gco2D_BatchBlit, gco2D_Blit, gco2D_Clear, gco2D_ColorLine, gco2D_ConstructColorBrush, gco2D_FilterBlitEx2, gco2D_FlushBrush, gco2D_Line, gco2D_LoadColorBrush, gco2D_LoadSolidBrush, gco2D_MonoBlit, gco2D_SetColorSourceAdvanced, gco2D_SetColorSourceEx, gco2D_SetGenericSource, gco2D_SetGenericTarget, gco2D_SetMaskedSourceEx, gco2D_SetSourceTileStatus, gco2D_StretchBlit, gcoSURF_Construct, gcoSURF_GetFormat, gcoSURF_SetBuffer.

gceSURF_FORMAT String Values	**Numeric**	**Description**
gcvSURF_UNKNOWN	0	Unknown format
gcvSURF_INDEX1	100	Palettized formats
gcvSURF_INDEX4	101	
gcvSURF_INDEX8	102	
gcvSURF_A2R2G2B2	200	RGB formats
gcvSURF_R3G3B2	201	
gcvSURF_A8R3G3B2	202	
gcvSURF_X4R4G4B4	203	
gcvSURF_A4R4G4B4	204	
gcvSURF_R4G4B4A4	205	
gcvSURF_X1R5G5B5	206	
gcvSURF_A1R5G5B5	207	
gcvSURF_R5G5B5A1	208	
gcvSURF_R5G6B5	209	
gcvSURF_R8G8B8	210	
gcvSURF_X8R8G8B8	211	
gcvSURF_A8R8G8B8	212	
gcvSURF_R8G8B8A8	213	

gcvSURF_G8R8G8B8	214	
gcvSURF_R8G8B8G8	215	
gcvSURF_X2R10G10B10	216	
gcvSURF_A2R10G10B10	217	
gcvSURF_X12R12G12B12	218	
gcvSURF_A12R12G12B12	219	
gcvSURF_X16R16G16B16	220	
gcvSURF_A16R16G16B16	221	
gcvSURF_A32R32G32B32	222	
gcvSURF_R8G8B8X8	223	
gcvSURF_R5G5B5X1	224	
gcvSURF_R4G4B4X4	225	
gcvSURF_A4B4G4R4	300	BGR formats
gcvSURF_A1B5G5R5	301	
gcvSURF_B5G6R5	302	
gcvSURF_B8G8R8	303	
gcvSURF_B16G16R16	304	
gcvSURF_X8B8G8R8	305	
gcvSURF_A8B8G8R8	306	
gcvSURF_A2B10G10R10	307	
gcvSURF_X16B16G16R16	308	
gcvSURF_A16B16G16R16	309	
gcvSURF_B32G32R32	310	
gcvSURF_X32B32G32R32	311	
gcvSURF_A32B32G32R32	312	
gcvSURF_B4G4R4A4	313	
gcvSURF_B5G5R5A1	314	
gcvSURF_B8G8R8X8	315	
gcvSURF_B8G8R8A8	316	
gcvSURF_X4B4G4R4	317	
gcvSURF_X1B5G5R5	318	
gcvSURF_B4G4R4X4	319	
gcvSURF_B5G5R5X1	320	
gcvSURF_X2B10G10R10	321	
gcvSURF_DXT1	400	Compressed formats
gcvSURF_DXT2	401	
gcvSURF_DXT3	402	
gcvSURF_DXT4	403	
gcvSURF_DXT5	404	
gcvSURF_CXV8U8	405	
gcvSURF_ETC1	406	
gcvSURF_YUY2	500	YUV formats
gcvSURF_UYVY	501	
gcvSURF_YV12	502	
gcvSURF_I420	503	
gcvSURF_NV12	504	
gcvSURF_NV21	505	
gcvSURF_NV16	506	
gcvSURF_NV61	507	
gcvSURF_YVYU	508	

gcvSURF_VYUY	509	
gcvSURF_D16	600	Depth formats
gcvSURF_D24S8	601	
gcvSURF_D32	602	
gcvSURF_D24X8	603	
gcvSURF_A4	700	Alpha formats
gcvSURF_A8	701	
gcvSURF_A12	702	
gcvSURF_A16	703	
gcvSURF_A32	704	
gcvSURF_A1	705	
gcvSURF_L4	800	Luminance formats
gcvSURF_L8	801	
gcvSURF_L12	802	
gcvSURF_L16	803	
gcvSURF_L32	804	
gcvSURF_L1	805	
gcvSURF_A4L4	900	Alpha / Luminance formats
gcvSURF_A2L6	901	
gcvSURF_A8L8	902	
gcvSURF_A4L12	903	
gcvSURF_A12L12	904	
gcvSURF_A16L16	905	
gcvSURF_L6V5U5	1000	Bump formats
gcvSURF_V8U8	1001	
gcvSURF_X8L8V8U8	1002	
gcvSURF_Q8W8V8U8	1003	
gcvSURF_A2W10V10U10	1004	
gcvSURF_V16U16	1005	
gcvSURF_Q16W16V16U16	1006	
gcvSURF_R8	1100	R / RG / RA formats
gcvSURF_X8R8	1101	
gcvSURF_G8R8	1102	
gcvSURF_X8G8R8	1103	
gcvSURF_A8R8	1104	
gcvSURF_R16	1105	
gcvSURF_X16R16	1106	
gcvSURF_G16R16	1107	
gcvSURF_X16G16R16	1108	
gcvSURF_A16R16	1109	
gcvSURF_R32	1110	
gcvSURF_X32R32	1111	
gcvSURF_G32R32	1112	
gcvSURF_X32G32R32	1113	
gcvSURF_A32R32	1114	
gcvSURF_RG16	1115	

gcvSURF_R16F	1200	Floating point formats
gcvSURF_X16R16F	1201	
gcvSURF_G16R16F	1202	
gcvSURF_X16G16R16F	1203	
gcvSURF_B16G16R16F	1204	
gcvSURF_X16B16G16R16F	1205	
gcvSURF_A16B16G16R16F	1206	
gcvSURF_R32F	1207	
gcvSURF_X32R32F	1208	
gcvSURF_G32R32F	1209	
gcvSURF_X32G32R32F	1210	
gcvSURF_B32G32R32F	1211	
gcvSURF_X32B32G32R32F	1212	
gcvSURF_A32B32G32R32F	1213	
gcvSURF_A16F	1214	
gcvSURF_L16F	1215	
gcvSURF_A16L16F	1216	
gcvSURF_A16R16F	1217	
gcvSURF_A32F	1218	
gcvSURF_L32F	1219	
gcvSURF_A32L32F	1220	
gcvSURF_A32R32F	1221	

2.6.17 gceSURF_GLOBAL_ALPHA_MODE Enumeration

Used in objects: gco2D_EnableAlphaBlend, gco2D_EnableAlphaBlendAdvanced, gcoSURF_EnableAlphaBlend.

gceSURF_GLOBAL_ALPHA_MODE String Values	Numeric	Description
gcvSURF_GLOBAL_ALPHA_OFF	0	
gcvSURF_GLOBAL_ALPHA_ON	1	
gcvSURF_GLOBAL_ALPHA_SCALE	2	

2.6.18 gceSURF_MONOPACK Enumeration

Used in objects: gco2D_GetPackSize, gco2D_MonoBlit, gco2D_SetMonochromeSource, gco2D_SURF_Blit, gcoSURF_MonoBlit.

gceSURF_MONOPACK String Values	Numeric	Description
gcvSURF_PACKED8	0	Each 32-bit chunk is 8 pixels wide, which means that it defines 4 vertical lines of pixel mask.
gcvSURF_PACKED16	1	
gcvSURF_PACKED32	2	
gcvSURF_UNPACKED	3	

2.6.19 gceSURF_PIXEL_ALPHA_MODE Enumeration

Used in objects: gco2D_EnableAlphaBlend, gco2D_EnableAlphaBlendAdvanced, gcoSURF_EnableAlphaBlend.

gceSURF_PIXEL_ALPHA_MODE String Values	Numeric	Description
gcvSURF_PIXEL_ALPHA_STRAIGHT	0	
gcvSURF_PIXEL_ALPHA_INVERSED	1	

2.6.20 gceSURF_PIXEL_COLOR_MODE Enumeration

Used in objects: gco2D_EnableAlphaBlend, gcoSURF_EnableAlphaBlend.

gceSURF_PIXEL_COLOR_MODE String Values	Numeric	Description
gcvSURF_COLOR_STRAIGHT	0	
gcvSURF_COLOR_MULTIPLY	1	

2.6.21 gceSURF_ROTATION Enumeration

Used in objects: gco2D_FilterBlitEx2, gco2D_SetColorSourceAdvanced, gco2D_SetColorSourceEx, gco2D_SetGenericSource, gco2D_SetGenericTarget, gco2D_SetMaskedSourceEx, gco2D_SetTargetEx, gcoSURF_Rotate.

gceSURF_ROTATION String Values	Numeric	Description
gcvSURF_0_DEGREE	0	
gcvSURF_90_DEGREE	1	
gcvSURF_180_DEGREE	2	
gcvSURF_270_DEGREE	3	
gcvSURF_FLIP_X	4	
gcvSURF_FLIP_Y	5	

2.6.22 gceSURF_TRANSPARENCY Enumeration

Used in objects: gco2D_SetColorSourceEx, gco2D_SetMonochromeSource, gcoSURF_Blit, gcoSURF_MonoBlit.

gceSURF_TRANSPARENCY String Values	Numeric	Description
gcvSURF_OPAQUE	0	each pixel of the bitmap overwrites the destination
gcvSURF_SOURCE_MATCH	1	source pixels compared against register value
gcvSURF_SOURCE_MASK	2	monochrome source mask defines transparency
gcvSURF_PATTERN_MASK	3	pattern mask defines transparency

2.6.23 gceSURF_TYPE Enumeration

Used in objects: gcoSURF_Construct, gcoSURF_GetFormat, gcoSURF_SetBuffer.

gceSURF_TYPE String Values	Numeric	Description
gcvSURF_TYPE_UNKNOWN	0	

gcvSURF_INDEX	1	
gcvSURF_VERTEX	2	
gcvSURF_TEXTURE	3	
gcvSURF_RENDER_TARGET	4	
gcvSURF_DEPTH	5	
gcvSURF_BITMAP	6	
gcvSURF_TILE_STATUS	7	
gcvSURF_IMAGE	8	
gcvSURF_MASK	9	
gcvSURF_SCISSOR	10	
gcvSURF_HIERARCHICAL_DEPTH	11	
gcvSURF_NUM_TYPES	12	Ensure that this is the last one!
		Combinations
gcvSURF_NO_TILE_STATUS	0x100	
gcvSURF_NO_VIDMEM	0x200	Used to allocate surfaces with no underlying vidmem node.
		In Android, vidmem node is allocated by another process.
gcvSURF_CACHEABLE	0x400	Used to allocate a cacheable surface.
gcvSURF_FLIP	0x800	If gcdANDROID_UNALIGNED_LINEAR_ COMPOSITION_ADJUST, the Resolve Target will be flip resolve from RT.
gcvSURF_RENDER_TARGET_NO_TILE_STATUS		= gcvSURF_RENDER_TARGET | gcvSURF_NO_TILE_STATUS
gcvSURF_DEPTH_NO_TILE_STATUS		= gcvSURF_DEPTH | gcvSURF_NO_TILE_STATUS
		Supported surface types with no vidmem node.
gcvSURF_BITMAP_NO_VIDMEM		= gcvSURF_BITMAP | gcvSURF_NO_VIDMEM
gcvSURF_TEXTURE_NO_VIDMEM		= gcvSURF_TEXTURE | gcvSURF_NO_VIDMEM
		Cacheable surface types with no vidmem node.
gcvSURF_CACHEABLE_BITMAP_NO_VIDMEM		= gcvSURF_BITMAP_NO_VIDMEM | gcvSURF_CACHEABLE
gcvSURF_CACHEABLE_BITMAP		= gcvSURF_BITMAP | gcvSURF_CACHEABLE
gcvSURF_FLIP_BITMAP		if gcdANDROID_UNALIGNED_LINEAR_ COMPOSITION_ADJUST, = gcvSURF_BITMAP | gcvSURF_FLIP

2.6.24 gceTILING Enumeration

Used in objects: gco2D_FilterBlitEx2, gco2D_SetGenericSource, gco2D_SetGenericTarget, gco2D_SetMaskedSourceEx.

gceTILING String Values	Numeric	Description
gcvLINEAR	0	
gcvTILED	1	4x4 tiles
gcvSUPERTILED	2	64x64 supertiles
gcvMULTI_TILED	3	Multi-pipe split tiles
gcvMULTI_SUPERTILED	4	Multi-pipe split supertiles
gcvMINORTILED	5	

2.7 2D Enumerations not specifically called

2.7.1 gce2D_PATTERN Enumeration

gce2D_PATTERN String Values	Numeric	Description
gcv2D_PATTERN_SOLID	0	
gcv2D_PATTERN_MONO	1	
gcv2D_PATTERN_COLOR	2	
gcv2D_PATTERN_INVALID	3	

2.7.2 gce2D_SOURCE Enumeration

gce2D_SOURCE String Values	Numeric	Description
gcv2D_SOURCE_MASKED	0	
gcv2D_SOURCE_MONO	1	
gcv2D_SOURCE_COLOR	2	
gcv2D_SOURCE_INVALID	3	

2.7.3 gceSURF_COLOR_TYPE Enumeration

gceSURF_COLOR_TYPE String Values	Numeric	Description
gcvSURF_COLOR_UNKNOWN	0	
gcvSURF_COLOR_LINEAR	1	
gcvSURF_COLOR_ALPHA_PRE	2	

2.8 Structures

2.8.1 gcs2D_PROFILE Structure

Used in objects: gco2D_ProfileEngine.

gcs2D_PROFILE Members	Type	Description
cycleCount	gctUINT32	32-bit counter incremented every 2D clock cycle;

		wraps to 0 upon counter overflow.
pixelsRendered	gctUINT32	Number of pixels rendered by the 2D engine; resets to 0 every time it is read.

2.8.2 gcsPOINT Structure: Point Definition

Used in objects: gco2D_MonoBlit.

gcsPOINT Members	Type	Description
x	gctINT32	X origin for point
y	gctINT32	Y origin for point

2.8.3 gcsRECT Structure: Rectangle Definition

Used in objects: gco2D_BatchBlit, gco2D_Blit, gco2D_Clear, gco2D_ColorLine, gco2_FilterBlitEx2, gco2D_Line, gco2D_MonoBlit, gco2D_MultiSourceBlit, gco2D_SetClipping, gco2D_SetSource, gco2D_SetStretchRectFactors, gco2D_StretchBlit, gcoSURF_Blit, gcoSURF_FilterBlit, gcoSURF_Line, gcoSURF_MonoBlit, gcsRECT_Height, gcsRECT_IsEqual, gcsRECT_IsOfEqualSize, gcsRECT_Normalize, gcsRECT_Rotate, gcsRECT_Set, gcsRECT_Width.

gcsRECT Members	Type	Description
left	gctINT32	Left
top	gctINT32	Top
right	gctINT32	Right
bottom	gctINT32	Bottom

3 OS User Objects

Each process must create one gcoOS object and one gcoHAL object for each thread that needs asynchronous access to the HAL.

gcoOS_Construct

Description:
Constructs a new gcoOS object.

Syntax:

```
gceSTATUS
gcoOS_Construct (
        IN gctPOINTER        Context,
        OUT gcoOS *          Os

);
```

Parameters:

Context	Pointer to OS-specific context.
Os	Pointer to a variable that holds the gcoOS object pointer.

gcoOS_Destroy

Description:
Destroys a gcoOS object.

Syntax:
```
gceSTATUS
gcoOS_Destroy (
       IN gcoOS            Os
);
```

Parameters:

Os	Pointer to the gcoOS object that needs to be destroyed.

gcoOS_MemCmp

Description:
Verifies if two specified memory regions are equal.

Syntax:
```
gceSTATUS
gcoOS_MemCmp (
       IN gctCONST_POINTER   Memory1,
       IN gctCONST_POINTER   Memory2,
       IN gctSIZE_T          Bytes
);
```

Parameters:

Memory1	Pointer to the first memory region to compare.
Memory2	Pointer to the second memory region to compare.
Bytes	Number of bytes to compare.

Returns:
gcvSTATUS_OK if the memory regions match, or **gcvSTATUS_MISMATCH** if the memory regions do not match.

gcoOS_MemCopy

Description:

Performs a memory copy from one location to another location. The memory cannot be overlapped.

Syntax:

```
gceSTATUS
gcoOS_MemCopy (
      IN gctPOINTER       Destination,
      IN gctCONST_POINTER Source,
      IN gctSIZE_T        Bytes
);
```

Parameters:

Destination	Pointer to the destination of the memory copy.
Source	Pointer to the memory source you want to copy from.
Bytes	The amount of memory, in bytes, you want to copy.

gcoOS_MemFill

Description:

Performs a memory fill.

Syntax:

```
gceSTATUS
gcoOS_MemFill (
      IN gctPOINTER       Memory,
      IN gctUINT8         Filler,
      IN gctSIZE_T        Bytes
);
```

Parameters:

Memory	Pointer to the memory to fill.
Filler	Value to fill the memory with.
Bytes	The number of bytes you want to fill.

gcoOS_PrintStr

Description:
Appends a "printf" formatted string to a string buffer and adjusts the offset into the string buffer. Since there is no checking for a buffer overflow, so ensure that the string buffer is large enough.

Syntax:

```
gceSTATUS
gcoOS_PrintStr (
      IN gctSTRING         String,
      IN gctSIZE_T         StringSize,
      IN OUT gctUINT_PTR   Offset,
      IN gctCONST_STRING   Format

);
```

Parameters:

String	Pointer to the string buffer.
StringSize	Size of string.
Offset	**IN**: Pointer to a variable that holds the current offset into the string buffer. **OUT**: Pointer to a variable that receives the new offset into the string buffer pointed to by <String> after the formatted string pointed to by <Format> has been appended to it.
Format	Pointer to a “printf” style format to append to the string buffer pointed to by <String> at the position <Offset>. Variable number of arguments that will be used by <Format>.

gcoOS_StrCmp

Description:
Verifies if two specified strings are equal.

Syntax:
```
gceSTATUS
gcoOS_StrCmp (
      IN gctCONST_STRING   String1,
      IN gctCONST_STRING   String2
);
```

Parameters:

String1	Pointer to the first string to compare to the second specified string.
String2	Pointer to the second string to compare to the first specified string.

Returns:
gcvSTATUS_OK if the strings match
gcvSTATUS_LARGER if String1 > String2
gcvSTATUS_SMALLER if String1 < String2

gcoOS_StrLen

Description:
Computes the length of a specified string.

Syntax:
```
gceSTATUS
gcoOS_StrLen (
      IN gctCONST_STRING   String,
      OUT gctSIZE_T *      Length
);
```

Parameters:

String	Pointer to the specified string.
Length	Pointer to a variable that will receive the length, in bytes, of the specified string.

gcoOS_ZeroMemory

Description:

Fills the specified memory with zeros.

Syntax:

```
gceSTATUS
gcoOS_ZeroMemory (
      IN gctPOINTER       Memory,
      IN gctSIZE_T        Bytes
);
```

Parameters:

Memory	Pointer to the memory to fill.
Bytes	The number of bytes of memory to fill with zero.

4 HAL User Objects

gcoHAL_Commit

Description:

Commits the current command buffer to hardware and optionally waits until the hardware is finished.

Syntax:

```
gceSTATUS
gcoHAL_Commit(
      IN gcoHAL           Hal,
      IN gctBOOL          Stall
);
```

Parameters:

Hal	Pointer to the gcoHAL object.
Stall	**gcvTRUE** if the thread needs to wait until the hardware is finished executing the committed command buffer.

gcoHAL_Construct

Description:

Constructs a new gcoHAL object.

Syntax:

```
gceSTATUS
```

```
gcoHAL_Construct (
      IN gctPOINTER       Context,
      IN gcoOS            Os,
      OUT gcoHAL *        Hal
);
```

Parameters:

Context	Pointer to a context that can be used by the platform specific functions.
Os	Pointer to a gcoOS object.
Hal	Pointer to a variable that holds the gcoHAL object pointer.

gcoHAL_Destroy

Description:
Destroys a gcoHAL object.

Syntax:

```
gceSTATUS
gcoHAL_Destroy(
      IN gcoHAL           Hal
);
```

Parameters:

Hal	Pointer to the gcoHAL object that you want to destroy.

gcoHAL_Get2DEngine

Description:

Gets the pointer to the gco2D object.

Parameters:

```
gceSTATUS
gcoHAL_Get2DEngine(
      IN gcoHAL            Hal,
      OUT gco2D *          Engine
```

```
        );
```

Parameters:

Hal	Pointer to the gcoHAL object.
Engine	Pointer to a variable receiving the gco2D object pointer.

gcoHAL_GetHardwareType

Description:
Gets the HAL hardware type to the TLS (thread local storage).

Parameters:

```
        gceSTATUS
        gcoHAL_GetHardwareType (
                IN gcoHAL           Hal,
                OUT gceHARDWARE_TYPE *   HardwareType
        );
```

Parameters:

Hal	Pointer to the gcoHAL object.
HardwareType	Pointer to a variable that will hold the hardware type.

gcoHAL_IsFeatureAvailable

Description:
Verifies if the specified feature is available in hardware.

Parameters:

```
        gceSTATUS
        gcoHAL_IsFeatureAvailable (
                IN gcoHAL           Hal,
                IN gceFEATURE       Feature
        );
```

Parameters:

Hal	Pointer to the gcoHAL object.
Feature	Feature to be verified.

gcoHAL_MapUserMemory

Description:
Maps a contiguous memory to GPU address space. **gcoHAL_MapUserMemory/gcoHAL_UnmapUserMemory** explicitly converts either CPU logical address or CPU physical address to the corresponding GPU address.

Performance:
- Extremely fast if CPU physical address of the user memory can be provided and the user memory is contiguous and inside GPU physical address or static virtual mapping space.
- Relatively fast if only CPU logical address of the user memory is provided and the user memory is contiguous and inside GPU physical address or static virtual mapping space.
- Slow if the user memory is not contiguous or not fully inside GPU physical address or static virtual mapping space.

Notes:
- If possible, try to avoid unnecessary/frequent calling of both APIs, especially for the slow conversion cases inside per-draw operation.
- These two APIs will use the current hardware type. Ensure that the correct hardware type is set before you call them.
- If Physical is valid, Logical will be ignored and memory will be considered contiguous.

Syntax:
```
gceSTATUS
gcoHAL_MapUserMemory (
      IN gctPOINTER       Logical
      IN gctUINT32        Physical
      IN gctSIZE_T        Size,
      OUT gctPOINTER *    Info,
      OUT gctUINT32_PTR   GPUAddress
);
```

Parameters:

Logical	Logical address of this memory
Physical	Physical address of this memory (optional) If not known or not used, please input gcvINVALID_ADDRESS
Size	Size in bytes of the memory to map

Info	Information record returned by gcoHAL_MapUserMemory
GPUAddress	GPU address returned by gcoHAL_MapUserMemory

Return Values:

The following status returns are supported:

gcvSTATUS_GENERIC_IO
gcvSTATUS_HEAP_CORRUPTED
gcvSTATUS_INTERFACE_ERROR
gcvSTATUS_INVALID_ADDRESS
gcvSTATUS_INVALID_ARGUMENT
gcvSTATUS_NOT_SUPPORTED
gcvSTATUS_OK
gcvSTATUS_OUT_OF_MEMORY
gcvSTATUS_OUT_OF_RESOURCES
gcvSTATUS_TIMEOUT

gcoHAL_SetHardwareType

Description:

Sets the HAL hardware type to the TLS (thread local storage).

Parameters:

```
gceSTATUS
gcoHAL_SetHardwareType (
        IN gcoHAL           Hal,
        IN gceHARDWARE_TYPE  HardwareType
);
```

Parameters:

Hal	Pointer to the gcoHAL object.
HardwareType	Hardware type.

gcoHAL_UnmapUserMemory

Description:
Unmaps a contiguous memory from GPU address space.

Syntax:
```
gceSTATUS
gcoHAL_UnmapUserMemory (
      IN gctPOINTER       Logical
      IN gctSIZE_T        Size,
      IN gctPOINTER      Info,
      IN gctUINT32   GPUAddress
);
```

Parameters:

Logical	Pointer to CPU logical memory to unmap
Size	Size in bytes of the memory to unmap
Info	Information record returned by gcoHAL_MapUserMemory
GPUAddress	GPU address returned by gcoHAL_MapUserMemory

Return values:
Same as gcoHAL_MapUserMemory.

5 2D User Objects

gco2D_BatchBlit

Description:
Generic blit for a batch of (source, destination) rectangle pairs.

Syntax:
```
gceSTATUS
gco2D_BatchBlit (
      IN gco2D          Engine,
      IN gctUINT32      RectCount,
      IN gcsRECT_PTR    SrcRect,
      IN gcsRECT_PTR    DestRect,
      IN gctUINT8       FgRop,
      IN gctUINT8       BgRop,
      IN gceSURF_FORMAT DestFormat
);
```

Parameters

Engine	Pointer to a gco2D object.
RectCount	The number of rectangles to draw. The array of rectangle positions to which the SrcRect and DestRect parameters point must have at least RectCount positions.
SrcRect	Points to an array of positions in (x0, y0)-(x1, y1) format.
DestRect	Points to an array of positions in (x0, y0)-(x1, y1) format.
FgRop	Foreground ROP to use with opaque pixels.
BgRop	Background ROP to use with transparent pixels.
DestFormat	The format of the destination buffer.

gco2D_Blit

Description:

Generic blit.

Syntax:

```
gceSTATUS
gco2D_Blit (
      IN gco2D            Engine,
      IN gctUINT32        RectCount,
      IN gcsRECT_PTR      Rect,
      IN gctUINT8         FgRop,
      IN gctUINT8         BgRop,
      IN gceSURF_FORMAT   DestFormat
);
```

Parameters:

Engine	Pointer to a gco2D object.
RectCount	The number of rectangles to draw. The array of line positions, pointed to by the Position parameter, must have at least RectCount positions.
Rect	Points to an array of positions in (x0, y0)-(x1, y1) format.
FgRop	The foreground ROP to use with opaque pixels.
BgRop	The background ROP to use with transparent pixels.
DestFormat	The format of the destination buffer.

gco2D_CalcStretchFactor

Description:
Calculates the stretch factors based on the sizes.

Syntax
```
gceSTATUS
gco2D_CalcStretchFactor(
      IN gco2D            Engine,
      IN gctINT32         SrcSize,
      IN gctINT32         DestSize,
      OUT gctUINT32_PTR   Factor
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcSize	Source size for horizontal or vertical direction.
DestSize	Destination size for horizontal or vertical direction.
Factor	Stretch factor in 16.16 fixed point format.

gco2D_Clear

Description:
Clears one or more rectangular areas. The color is specified in A8R8G8B8 format.

Syntax
```
gceSTATUS
gco2D_Clear (
      IN gco2D            Engine,
      IN gctUINT32        RectCount,
      IN gcsRECT_PTR      Rect,
      IN gctUINT32        Color32,
      IN gctUINT8         FgRop,
      IN gctUINT8         BgRop,
      IN gceSURF_FORMAT   DestFormat
);
```

Parameters:

Engine	Pointer to a gco2D object.
RectCount	The number of rectangles to draw. The array of rectangle positions, to which the Position parameter points, must have at least RectCount positions.
Rect	Points to an array of positions in (x0, y0)-(x1, y1) format.
Color32	A8R8G8B8 clear color value.
FgRop	Foreground ROP to use with opaque pixels.
BgRop	Background ROP to use with transparent pixels.
DestFormat	The format of the destination buffer.

gco2D_ColorLine

Description:
Draws one or more Bresenham lines with a given color for each line.

Syntax:
```
gceSTATUS
gco2D_ColorLine (
      IN gco2D           Engine,
      IN gctUINT32       LineCount,
      IN gcsRECT_PTR     Position,
      IN gctUINT32       Color32,
      IN gctUINT8        FgRop,
      IN gctUINT8        BgRop,
      IN gceSURF_FORMAT  DestFormat
);
```

Parameters:

Engine	Pointer to a gco2D object.

LineCount	The number of lines to draw. The array, to which the Position parameter points, must have at least LineCount positions.
Position	Points to an array of positions in (x0, y0)-(x1, y1) format.
Color32	Source color array in A8R8B8G8 format.
FgRop	The foreground ROP to use with opaque pixels.
BgRop	The background ROP to use with transparent pixels.
DestFormat	The format of the destination buffer.

gco2D_Construct

Description:
Constructs a new gco2D object.

Syntax:
```
gceSTATUS
gco2D_Construct (
      IN gcoHAL           Hal,
      OUT gco2D *         Engine
);
```

Parameters:

Hal	Pointer to a gcoHAL object.
Engine	Pointer to a variable that holds the pointer to the gco2D object.

gco2D_ConstructColorBrush

Description:
Creates a color gcoBRUSH object.

Syntax:
```
gceSTATUS
gco2D_ConstructColorBrush (
      IN gco2D            Engine,
      IN gctUINT32        OriginX,
      IN gctUINT32        OriginY,
      IN gctPOINTER       Address,
      IN gceSURF_FORMAT   Format,
      IN gctUINT64        Mask,
      OUT gcoBRUSH *      Brush
);
```

Parameters:

Engine	Pointer to a gco2D object.
OriginX **OriginY**	(X, Y) origin of the pattern in range 0 to 7.
Address	The location of the pattern bitmap in system memory.
Format	The format of the source bitmap.
Mask	Each 64 bits of mask corresponds to one pixel of the 8x8 pattern. Each pattern bit is used to determine transparency of the corresponding pixel. That is, each mask bit selects between foreground and background ROPs. If the bit is 0, the background ROP is used; If the bit is 1, the foreground ROP is used. The mask mapping is:

7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

Brush	Pointer to the variable that holds the gcoBRUSH object pointer.

gco2D_ConstructMonochromeBrush

Description:

Creates a new monochrome gcoBRUSH object.

Syntax:

```
gceSTATUS
gco2D_ConstructMonochromeBrush (
      IN gco2D          Engine,
      IN gctUINT32      OriginX,
      IN gctUINT32      OriginY,
      IN gctUINT32      ColorConvert,
      IN gctUINT32      FgColor,
      IN gctUINT32      BgColor,
      IN gctUINT64      Bits,
      IN gctUINT64      Mask,
      OUT gcoBRUSH *    Brush
);
```

Parameters:

Engine	Pointer to a gco2D object.
OriginX/Y	Specify the X and Y orgins of the pattern in range 0 to 7.
ColorConvert	The values of FgColor and BgColor parameters are stored direcltly in internal color registers and are used either directly to initialize a pattern or, if ColorConvert is not zero, are converted to the destination format prior to use.
FgColor/BgColor	Foreground and background colors of the pattern. The values are used to initialize the 8x8 pattern. If the values are in destination format, set ColorConvert to 0; otherwise, provide the values in ARGB8 format and set ColorConvert to 1 to instruct the hardware to convert the values to the destination format before use.
Bits	64 bits of pixel bits. Each bit represents one pixel and is used to choose between foreground and background colors. If the bit is 0, the background color is used; otherwise, the foreground color is used. The mapping between Bits parameter and the actual pattern pixels is the same as of the Mask parameter.
Mask	64 bits of mask, where each bit corresponds to one pixel of 8x8 pattern. Each bit of the mask is used to determine transparency of the corresponding pixel. In other words, each mask bit is used to select between foreground and background ROPs. If the bit is 0, background ROP is used on the pixel; if 1, the foreground ROP is used. The mapping between Mask parameter bits and the actual

	pattern pixels is as follows: 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 23 22 21 20 19 18 17 16 31 30 29 28 27 26 25 24 39 38 37 36 35 34 33 32 47 46 45 44 43 42 41 40 55 54 53 52 51 50 49 48 63 62 61 60 59 58 57 56
Brush	Pointer to the variable that holds the gcoBRUSH object pointer.

gco2D_ConstructSingleColorBrush

Description:

Creates a new solid color gcoBRUSH object.

Syntax:

```
gceSTATUS
gco2D_ConstructSingleColorBrush (
      IN gco2D          Engine,
      IN gctUINT32      ColorConvert,
      IN gctUINT32      Color,
      IN gctUINT64      Mask,
      OUT gcoBRUSH *    Brush
);
```

Parameters:

Engine	Pointer to a gco2D object.
ColorConvert	The Color parameter value is stored direcltly in the internal color registers and is used either directly to initialize a pattern or, if ColorConvert is not zero, it is converted to the destination format prior to use.
Color	The color value of the pattern. The value is used to initialize an 8x8 pattern. If the value is in destination format, set ColorConvert to 0; otherwise, provide the value in ARGB8 format and set ColorConvert to 1 to instruct the hardware to convert the value to the detination format prior to use.
Mask	64 bits of mask, where each bit corresponds to one pixel of 8x8 pattern. Each bit of the mask is used to determine transparency of the corresponding pixel. In other words, each mask bit is used to select between foreground and background ROPs. If the bit is 0, background ROP is used on the pixel; if 1, the foreground ROP is used. The mapping between Mask parameter bits and actual pattern pixels is as follows:

7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

Brush	Pointer to a variable that holds the gcoBRUSH object pointer.

gco2D_Destroy

Description:
Destroys a gco2D object.

Syntax:
```
gceSTATUS
gco2D_Destroy (
      IN gco2D            Engine
);
```

Parameters:

Engine	Pointer to the gco2D object that you want to destroy.

gco2D_DisableAlphaBlend

Description:
Disables alpha blending engine in the hardware and engage the ROP engine.

Syntax:
```
gceSTATUS
gco2D_DisableAlphaBlend (
      IN gco2D            Engine
);
```

Parameters:

Engine	Pointer to a gco2D object.

gco2D_EnableAlphaBlend

Description:

Enables alpha blending engine in the hardware and disengages the ROP engine. Use this function with hardware which has an older PixelEngine. Use **gco2D_EnableAlphaBlendAdvanced** with PixelEngine 2.0.

Syntax:

```
gceSTATUS
gco2D_EnableAlphaBlend (
    IN gco2D                        Engine,
    IN gctUINT8                     SrcGlobalAlphaValue,
    IN gctUINT8                     DstGlobalAlphaValue,
    IN gceSURF_PIXEL_ALPHA_MODE     SrcAlphaMode,
    IN gceSURF_PIXEL_ALPHA_MODE     DstAlphaMode,
    IN gceSURF_GLOBAL_ALPHA_MODE    SrcGlobalAlphaMode,
    IN gceSURF_GLOBAL_ALPHA_MODE    DstGlobalAlphaMode,
    IN gceSURF_BLEND_FACTOR_MODE    SrcFactorMode
    IN gceSURF_BLEND_FACTOR_MODE    DstFactorMode
    IN gceSURF_PIXEL_COLOR_MODE     SrcColorMode,
    IN gceSURF_PIXEL_COLOR_MODE     DstColorMode
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcGlobalAlphaValue **DstGlobalAlphaValue**	Source and destination global alpha values for the color components.
SrcAlphaMode **DstAlphaMode**	Source and destination per-pixel alpha component mode.
SrcGlobalAlphaMode **DstGlobalAlphaMode**	Source and destination global per-pixel alpha values selection.
SrcFactorMode **DstFactorMode**	Source and destination final blending factor mode.
SrcColorMode **DstColorMode**	Source and destination per-pixel color component mode.

gco2D_EnableAlphaBlendAdvanced

Description:

Enables alpha blending engine in the hardware and disengages the ROP engine. Use this function with hardware that uses PixelEngine 2.0. Use **gco2D_EnableAlphaBlend** with hardware which has an older PixelEngine.

Syntax:

```
gceSTATUS
gco2D_EnableAlphaBlendAdvanced (
    IN gco2D                        Engine,
    IN gceSURF_PIXEL_ALPHA_MODE     SrcAlphaMode,
    IN gceSURF_PIXEL_ALPHA_MOD      DstAlphaMode,
    IN gceSURF_GLOBAL_ALPHA_MODE    SrcGlobalAlphaMode,
    IN gceSURF_GLOBAL_ALPHA_MOD     DstGlobalAlphaMode,
    IN gceSURF_BLEND_FACTOR_MODE    SrcFactorMode
    IN gceSURF_BLEND_FACTOR_MODE    DstFactorMode
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcAlphaMode **DstAlphaMode**	Source and destination per-pixel alpha component mode.
SrcGlobalAlphaMode **DstGlobalAlphaMode**	Source and destination global per-pixel alpha values selection.
SrcFactorMode **DstFactorMode**	Source and destination final blending factor mode.

gco2D_EnableDither

Description:

Enables or disables dithering.

Syntax:

```
gceSTATUS
```

```
gco2D_EnableDither(
    IN gco2D              Engine,
    IN gctBOOL            Enable
);
```

Parameters

Engine	Pointer to a gco2D object.
Enable	**gcvTRUE** to enable dithering, **gcvFALSE** to disable.

gco2D_EnableUserFilterPasses

Description:
Selects the pass(es) to be done for user-defined filter.

Syntax:

```
gceSTATUS
gco2D_EnableUserFilterPasses (
      IN gco2D            Engine,
      IN gctBOOL          HorPass,
      IN gctBOOL          VerPass
);
```

Parameters:

Engine	Pointer to a gco2D object.
HorPass	Enable horizontal pass filter if HorPass is **gcvTRUE**; otherwise, disable this pass.
VerPass	Enable vertical pass filter if VerPass is **gcvTRUE**; otherwise, disable this pass.

gco2D_FilterBlitEx2

Description:
Filter blit. If the output format is multi planar YUV, do only color conversion.

Syntax:

```
gceSTATUS
gco2D_FilterBlitEx2(
      IN gco2D                Engine,
      IN gctUINT32_PTR        SrcAddresses,
      IN gctUINT32            SrcAddressNum,
      IN gctUINT32_PTR        SrcStrides,
      IN gctUINT32            SrcStrideNum,
      IN gceTILING            SrcTiling,
      IN gceSURF_FORMAT       SrcFormat,
      IN gceSURF_ROTATION     SrcRotation,
      IN gctUINT32            SrcSurfaceWidth,
      IN gctUINT32            SrcSurfaceHeight,
      IN gcsRECT_PTR          SrcRect,
      IN gctUINT32_PTR        DestAddresses,
      IN gctUINT32            DestAddressNum,
      IN gctUINT32_PTR        DestStrides,
      IN gctUINT32            DestStrideNum,
      IN gceTILING            DestTiling,
      IN gceSURF_FORMAT       DestFormat,
      IN gceSURF_ROTATION     DestRotation,
      IN gctUINT32            DestSurfaceWidth,
      IN gctUINT32            DestSurfaceHeight,
      IN gcsRECT_PTR          DestRect,
      IN gcsRECT_PTR          DestSubRect
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcAddresses	GPU address array of the source surface for different color channels according to the requirements of SrcFormat and SrcTiling.
SrcAddressNum	Number of SrcAddresses.
SrcStrides	Stride array of the source surface in bytes for different color channels according to the the requirements of SrcFormat and SrcTiling.
SrcStrideNum	Number of SrcStrides.
SrcTiling	The tiling mode of the source surface.
SrcFormat	Format of the source surface.
SrcRotation	Specifies the source surface rotation angle.
SrcSurfaceWidth	The width in pixels of the source surface.
SrcSurfaceHeight	The height in pixels of the source surface for the rotaion in PE 2.0.
SrcRect	Coordinates of the entire source image.
DestAddresses	GPU address array of the source surface for different color channels according to the requirements of DestFormat and DestTiling.
DestAddressNum	Number of DestAddresses.
DestStrides	Stride array of the destination surfaces in bytes for different color channels according to the requirement of DestFormat and DestTiling.
DestStrideNum	Number of DestStrides.
DestTiling	The tiling mode of the destination surface.
DestFormat	Format of the destination surface.
DestRotation	Specifies the destination surface rotation angle.

DestSurfaceWidth	The width in pixels of the destination surface.
DestSurfaceHeight	The height in pixels of the destination surface for the rotation in PE 2.0.
DestRect	Coordinates of the entire destination image.
DestSubRect	Coordinates of a sub area within the destination to render. If DestSubRect is **gcvNULL**, the complete image will be rendered using coordinates set by DestRect. If DestSubRect is not **gcvNULL** and DestSubRect and DestRect are not equal, DestSubRect is assumed to be within DestRect and will be used to render the sub area only.

gco2D_Flush

Description:

Flushes the 2D pipeline.

Syntax:

```
gceSTATUS
gco2D_Flush (
      IN gco2D            Engine
);
```

Parameters:

Engine	Pointer to a gco2D object.

gco2D_FlushBrush

Description:

Sets the maximum number of brushes in the brush cache.

Syntax:

```
gceSTATUS
gco2D_FlushBrush(
      IN gco2D            Engine,
      IN gcoBRUSH         Brush,
      IN gceSURF_FORMAT   Format
);
```

Parameters:

Engine	Pointer to a gco2D object.
Brush	Pointer to a valid gcoBRUSH object.
Format	Format for destination surface when using color conversion.

gco2D_FreeFilterBuffer

Description:
Frees the temporary buffer that was allocated by the filter blit operation.

Syntax:
```
gceSTATUS
gco2D_FreeFilterBuffer (
      IN gco2D             Engine
);
```

Parameters:

Engine	Pointer to a gco2D object.

gco2D_GetBrushCache

Description:
Returns a pointer to the brush cache.

Syntax:
```
gceSTATUS
gco2D_GetBrushCache (
      IN gco2D                    Engine,
      IN OUT gcoBRUSH_CACHE *     BrushCache
);
```

Parameters:

Engine	Pointer to a gco2D object.
BrushCache	Pointer to a gcoBRUSH_CACHE object.

gco2D_GetPackSize

Description:

Retrieves monochrome stream pack size.

Syntax:

```
gceSTATUS
gco2D_GetPackSize (
       IN gceSURF_MONOPACK StreamPack,
       OUT gctUINT32 *     PackWidth,
       OUT gctUINT32 *     PackHeight
);
```

Parameters:

StreamPack	Stream pack code.
PackWidth	Monochrome stream pack width.
PackHeight	Monochrome stream pack height.

gco2D_Line

Description:

Draws one or more Bresenham lines with a given brush.

Syntax:

```
gceSTATUS
gco2D_Line (
       IN gco2D           Engine,
       IN gctUINT32       LineCount,
       IN gcsRECT_PTR     Position,
       IN gcoBRUSH        Brush,
       IN gctUINT8        FgRop,
       IN gctUINT8        BgRop
       IN gceSURF_FORMAT  DestFormat
);
```

Parameters:

Engine	Pointer to a gco2D object.
LineCount	The number of lines to draw. The array of line positions to which the Position parameter points must have at least LineCount positions.

Position	Points to an array of positions in (x0, y0)-(x1, y1) format.
Brush	The brush to use for drawing.
FgRop	The foreground ROP to use with opaque pixels.
BgRop	The background ROP to use with transparent pixels.
DestFormat	The format of the destination buffer.

gco2D_LoadColorBrush

Description:
Creates a color brush object.

Syntax:
```
gceSTATUS
gco2D_LoadColorBrush (
      IN gco2D            Engine,
      IN gctUINT32        OriginX,
      IN gctUINT32        OriginY,
      IN gctUINT32        Address,
      IN gceSURF_FORMAT   Format,
      IN gctUINT64        Mask,
);
```

Parameters:

Engine	Pointer to a gco2D object.
OriginX	Specify the pattern origin in range 0 to 7.
OriginY	Specify the pattern origin in range 0 to 7.
Address	The location of the pattern bitmap in the system memory.
Format	The format of the source bitmap.

Mask 64 bits of mask, where each bit corresponds to one pixel of 8x8 pattern. Each bit of the mask is used to determine transparency of the corresponding pixel. In other words, each mask bit is used to select between foreground and background ROPs. If the bit is 0, background ROP is used on the pixel. If the bit is 1, the foreground ROP is used. The mapping between Mask parameter bits and actual pattern pixels is as follows:

7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

gco2D_LoadMonochromeBrush

Description:

Allows quick set of a new monochrome brush object.

Syntax:

```
gceSTATUS
gco2D_LoadMonochromeBrush (
      IN gco2D          Engine,
      IN gctUINT32      OriginX,
      IN gctUINT32      OriginY,
      IN gctUINT32      ColorConvert,
      IN gctUINT32      FgColor,
      IN gctUINT32      BgColor,
      IN gctUINT64      Bits,
      IN gctUINT64      Mask,

);
```

Parameters:

Engine	Pointer to a gco2D object.
OriginX	Specify the pattern origin in range 0 to 7.
OriginY	Specify the pattern origin in range 0 to 7.
ColorConvert	The values of FgColor and BgColor parameters are stored directly in internal color registers. The parameters are used either directly to initialize pattern or converted to the destination format before it is

	actually used. The later happens if ColorConvert is not zero.
FgColor	Foreground colors of the pattern. The value is used to initialize 8x8 pattern. If the values are in destination format, set ColorConvert to 0; otherwise, provide the values in ARGB8 format and set ColorConvert to 1 to instruct the hardware to convert the values to the format destination before they are actually used.
BgColor	Background colors of the pattern. The value is used to initialize 8x8 pattern. If the values are in destination format, set ColorConvert to 0; otherwise, provide the values in ARGB8 format and set ColorConvert to 1 to instruct the hardware to convert the values to the format destination before they are actually used.
Bits	64 bits of pixel bits. Each bit represents one pixel and is used to choose between foreground and background colors. If the bit is 0, the background color is used; otherwise, the foreground color is used. The mapping between Bits parameter and the actual pattern pixels is the same as of the Mask parameter.
Mask	64 bits of mask, where each bit corresponds to one pixel of 8x8 pattern. Each bit of the mask is used to determine transparency of the corresponding pixel. In other words, each mask bit is used to select between foreground and background ROPs. If the bit is 0, background ROP is used on the pixel; if 1, the foreground ROP is used. The mapping between Mask parameter bits and actual pattern pixels is as follows:

7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

gco2D_LoadPalette

Description:
Loads a 256-entry color table for INDEX8 source surfaces.

Syntax:
```
gceSTATUS
gco2D_LoadPalette (
      IN gco2D            Engine,
      IN gctUINT          FirstIndex,
      IN gctUINT          IndexCount,
      IN gctPOINTER       ColorTable,
      IN gctBOOL          ColorConvert
);
```

Parameters:

Engine	Pointer to a gco2D object.
FirstIndex	The index from which to start loading (0…255).
IndexCount	The number of indices to load (FirstIndex + IndexCount <= 256).
ColorTable	Pointer to the color table to load. The value of the pointer should be set to the first value to load no matter what the value of FirstIndex is. The table must consist of 32-bit entries that contain color values in either ARGB8 or the destination color format (see ColorConvert).
ColorConvert	If set to **gcvTRUE**, the 32-bit values in the table are assumed to be converted by the hardware to the destination format as needed. If set to **gcvFALSE**, the 32-bit values in the table are assumed to be preconverted to the destination format.

gco2D_LoadSolidBrush

Description:

Programs the specified solid color brush.

Syntax:

```
gceSTATUS
gco2D_LoadSolidBrush (
        IN gco2D            Engine,
        IN gceSURF_FORMAT   Format,
        IN gctUINT32        ColorConvert,
        IN gctUINT32        Color,
        IN gctUINT64        Mask
);
```

Parameters

Engine	Pointer to the gco2D object.
Format	Format for destination surface when using color conversion.
ColorConvert	The value of the Color parameter is stored directly in internal color register and is used either directly to initialize pattern or is converted to the format of destination before it is used. The later happens if ColorConvert is not zero.
Color	The color value of the pattern. The value will be used to initialize 8x8 pattern. If the value is in destination format, set ColorConvert to 0; otherwise, provide the value in ARGB8 format and set

	ColorConvert to 1 to instruct the hardware to convert the value to the destination format before it is actually used.
Mask	64 bits of mask, where each bit corresponds to one pixel of 8x8 pattern. Each bit of the mask is used to determine transparency of the corresponding pixel. In other words, each mask bit is used to select between foreground and background ROPs. If the bit is 0, background ROP is used on the pixel. If the bit is 1, the foreground ROP is used. The mapping between Mask parameter bits and actual pattern pixels is as follows:

7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

gco2D_MonoBlit

Description:

Monochrome blit.

Syntax:

```
gceSTATUS
gco2D_MonoBlit (
      IN gco2D            Engine,
      IN gctPOINTER       StreamBits,
      IN gcsPOINT_PTR     StreamSize,
      IN gcsRECT_PTR      StreamRect,
      IN gceSURF_MONOPACK SrcStreamPack,
      IN gceSURF_MONOPACK DstStreamPack,
      IN gcsRECT_PTR      DestRect,
      IN gctUINT32        FgRop,
      IN gctUINT32        BgRop,
      IN gceSURF_FORMAT   DestFormat
);
```

Parameters

Engine	Pointer to a gco2D object.
StreamBits	Pointer to monochrome bitmap.
StreamSize	Size of the monochrome bitmap in pixels.
StreamRect	Bounding rectangle of the area within the bitmap to render.

SrcStreamPack	Source bitmap packing.
DestStreamPack	Packing of the bitmap in the command stream.
DestRect	Pointer to an array of destination rectangles.
FgRop	Foreground ROP to use with opaque pixels.
BgRop	Background ROP to use with transparent pixels.
DestFormat	Destination surface format.

gco2D_ProfileEngine

Description:
Reads the profile registers available in the 2D engine and sets them in the profile. The pixelsRendered counter is reset to 0 after reading.

Syntax:
```
gceSTATUS
gco2D_ProfileEngine (
      IN gco2D                     Engine,
      OPTIONAL gcs2D_PROFILE_PTR   Profile
);
```

Parameters:

Engine	Pointer to a gco2D object.
Profile	Pointer to a gcs2D_Profile structure which contains two members: • cycleCount– 32-bit counter incremented every 2D clock cycle; wraps to 0 upon counter overflow. • pixelsRendered– Number of pixels rendered by the 2D engine; resets to 0 every time it is read.

gco2D_QueryU32

Description:
Queries 2D engine for unsigned 32 bit information.

Syntax:

```
gceSTATUS
gco2D_QueryU32 (
      IN gco2D            Engine,
      IN gce2D_QUERY      Item,
      OUT gctUINT32_PTR   Value
);
```

Parameters:

Engine	Pointer to the gco2D object.
Item	Item to query.
Value	Value for the queried item.

gco2D_SetAutoFlushCycles

Description:
Sets the GPU clock cycles after which the idle 2D engine will trigger a flush.

Syntax:

```
gceSTATUS
gco2D_SetAutoFlushCycles (
      IN gco2D            Engine,
      IN UINT32           Cycles
);
```

Parameters:

Engine	Pointer to a gco2D object.
Cycles	Number of GPU cycles to wait before triggering idle engine auto-flush.

gco2D_SetBitBlitMirror

Description:
Enables/disables 2D Bit BLT mirroring.

Syntax:

```
gceSTATUS
gco2D_SetBitBlitMirror (
      IN gco2D            Engine,
      IN gctBOOL          HorizontalMirror,
      IN gctBOOL          VerticalMirror
);
```

Parameters:

Engine	Pointer to a gco2D object.
HorizontalMirror	Horizontal mirror enable flag.
VerticalMirror	Vertical mirror enable flag.

Returns:

Returns gcvSTATUS_OK if successful.

gco2D_SetBrushLimit

Description:
Sets the maximum number of brushes in the cache.

Syntax:
```
gceSTATUS
gco2D_SetBrushLimit (
      IN gco2D            Engine,
      IN gctUINT          MaxCount
);
```

Parameters:

Engine	Pointer to a gco2D object.
MaxCount	The maximum number of brushes allowed in the cache at the same time.

gco2D_SetClipping

Description:
Sets clipping rectangle.

Syntax:
```
gceSTATUS
gco2D_SetClipping (
      IN gco2D            Engine,
      IN gcsRECT_PTR      Rect
);
```

Parameters:

Engine	Pointer to the gco2D object.

Rect	Pointer to a valid destination rectangle. The valid range is 0 to 32768. A pixel is valid if the following is true: (pixelX >= Left) && (pixelX < Right) && (pixelY >= Top) && (pixelY < Bottom)

gco2D_SetColorSourceAdvanced

Description:
Configures color source surface. The function is compatible with PixelEngine 2.0 and above.

Syntax:
```
gceSTATUS
gco2D_SetColorSourceAdvanced (
      IN gco2D            Engine,
      IN gctUINT32        Address,
      IN gctUINT32        Stride,
      IN gceSURF_FORMAT   Format,
      IN gceSURF_ROTATION Rotation,
      IN gctUINT32        SurfaceWidth,
      IN gctUINT32        SurfaceHeight,
      IN gctBOOL          CoordRelative
);
```

Parameters:

Engine	Pointer to the gco2D object.
Address	Source surface base address.
Stride	Source surface stride in bytes.
Format	Source surface color format.
Rotation	Type of rotation.
SurfaceWidth	Source surface width in pixels. Required only if the surface is rotated.
SurfaceHeight	Source surface height in pixels. Required only if the surface is rotated in Pixel Engine 2.0. Equal to the height of the source surface in pixels.
CoordRelative	If **gcvFALSE**, the source origin represents absolute pixel coordinate within the source surface. If **gcvTRUE**, the source origin represents the offset from the destination origin.

gco2D_SetColorSourceEx

Description:

Configures color source. The function is compatible with cores which have a PixelEngine older than version 2.0. Use **gco2D_SetColorSourceAdvanced** for cores with Pixel Engine 2.0.

Syntax:

```
gceSTATUS
gco2D_SetColorSourceEx (
      IN gco2D                Engine,
      IN gctUINT32            Address,
      IN gctUINT32            Stride,
      IN gceSURF_FORMAT       Format,
      IN gceSURF_ROTATION     Rotation
      IN gctUINT32            SurfaceWidth,
      IN gctUINT32            SurfaceHeight,
      IN gctBOOL              CoordRelative,
      IN gceSURF_TRANSPARENCY Transparency,
      IN gctUINT32            TransparencyColor
);
```

Parameters:

Engine	Pointer to the gco2D object.
Address	Source surface base address.
Stride	Source surface stride in bytes.
Format	Source surface color format.
Rotation	Type of rotation.
SurfaceWidth	Source surface width in pixels. Required only if the surface is rotated.
SurfaceHeight	Source surface height in pixels. Required only if the surface is rotated in Pixel Engine 2.0. Equal to the height of the source surface in pixels.
CoordRelative	If **gcvFALSE**, the source origin represents absolute pixel coordinate within the source surface. If **gcvTRUE**, the source origin represents the offset from the destination origin.
Transparency	The transparency simply comes down to selecting a ROP code to use. Opaque pixels use foreground ROP and transparent pixels use

	background ROP. **gcvSURF_OPAQUE** — each pixel of the bitmap overwrites the destination **gcvSURF_SOURCE_MATCH** — source pixels compared against register value **gcvSURF_SOURCE_MASK** — monochrome source mask defines transparency **gcvSURF_PATTERN_MASK** — pattern mask defines transparency
TransparencyColor	This value is used in **gcvSURF_SOURCE_MATCH** transparency mode. The value is compared against each pixel to determine transparency. If the values are equal, the pixel is transparent; otherwise, the pixel is opaque.

gco2D_SetCurrentSourceIndex

Description:
Supports multi-source.

Syntax:

```
gceSTATUS
gco2D_SetCurrentSourceIndex (
      IN gco2D           Engine,
      IN gctUINT32       SrcIndex
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcIndex	Current source index number of multi-source.

gco2D_SetFilterType

Description:
Sets the filter type.

Syntax:

```
gceSTATUS
gco2D_SetFilterType (
      IN gco2D           Engine,
      IN gceFILTER_TYPE  FilterType
);
```

Parameters:

Engine	Pointer to a gco2D object.
FilterType	Filter type for the filter blit.

gco2D_SetGdiStretchMode

Description:
Enables/disables 2D GDI stretch mode for integral multiple stretch.

Syntax:

```
gceSTATUS
gco2D_SetGdiStretchMode (
      IN gco2D            Engine,
      IN gctBOOL          Enable
);
```

Parameters:

Engine	Pointer to the gco2D object.
Enable	Enable/disable integral multiple stretch.

gco2D_SetKernelSize

Description:
Sets the kernel size.

Syntax:

```
gceSTATUS
gco2D_SetKernelSize (
      IN gco2D            Engine,
      IN gctUINT8         HorKernelSize,
      IN gctUINT8         VerKernelSize
);
```

Parameters:

Engine	Pointer to a gco2D object.
HorKernelSize	The kernel size for the horizontal pass.
VerKernelSize	The kernel size for the vertical pass.

gco2D_SetMaskedSourceEx

Description:

Configures masked color source.

Syntax:

```
gceSTATUS
gco2D_SetMaskedSourceEx (
      IN gco2D            Engine,
      IN gctUINT32        Address,
      IN gctUINT32        Stride,
      IN gceSURF_FORMAT   Format
      IN gctBOOL          CoordRelative,
      IN gceSURF_MONOPACK MaskPack,
      IN gceSURF_ROTATION Rotation,
      IN gctUINT32        SurfaceWidth,
      IN gctUINT32        SurfaceHeight

);
```

Parameters:

Engine	Pointer to the gco2D object.
Address	Source surface base address.
Stride	Source surface stride in bytes.
Format	Source surface color format.
CoordRelative	If **gcvFALSE**, the source origin represents absolute pixel coordinate within the source surface. If **gcvTRUE**, the source origin represents the offset from the destination origin.
MaskPack	Determines how many horizontal pixels there are in each 32-bit chunk of monochrome mask. For example, if set to **gcvSURF_PACKED8**, each 32-bit chunk is 8 pixels wide. This means that it defines 4 vertical lines of pixel mask.
Rotation	Type of rotation in PixelEngine 2.0.
SurfaceWidth	Source surface width in pixels. Required only if the surface is rotated in PixelEngine 2.0.
SurfaceHeight	Source surface height in pixels. Required only if the surface is rotated in Pixel Engine 2.0.

gco2D_SetMonochromeSource

Description:

Configures monochrome color source.

Syntax:

```
gceSTATUS
gco2D_SetMonochromeSource (
      IN gco2D            Engine,
      IN gctBOOL          ColorConvert,
      IN gctUINT8         MonoTransparency,
      IN gceSURF_MONOPACK DataPack,
      IN gctBOOL          CoordRelative,
      IN gceSURF_TRANSPARENCY Transparency,
      IN gctUINT32        FgColor,
      IN gctUINT32        BgColor
);
```

Parameters:

Engine	Pointer to the gco2D object.
ColorConvert	The values of FgColor and BgColor parameters are stored directly in the internal color registers and are used either directly as the source color or, if ColorConvert is **gcvTRUE**, converted to the format of the destination prior to being used.
MonoTransparency	This value is used in **gcvSURF_SOURCE_MATCH** transparency mode. The value can be either 0 or 1 and is compared against each mono-pixel to determine the transparency of the pixel. If the values are equal, the pixel is transparent; otherwise, the pixel is opaque.
DataPack	Determines how many horizontal pixels there are in each 32-bit chunk of monochrome bitmap. For example, if set to **gcvSURF_PACKED8**, each 32-bit chunk is 8 pixels wide. This means that it defines 4 vertical lines of pixels.
CoordRelative	If **gcvFALSE**, the source origin represents absolute pixel coordinate within the source surface. If **gcvTRUE**, the source origin represents the offset from the destination origin.
Transparency	Transparency is determined by the ROP code that is used. Opaque pixels use foreground ROP and transparent pixels use background ROP. **gcvSURF_OPAQUE** —each pixel of the bitmap overwrites the destination **gcvSURF_SOURCE_MATCH** —source pixels compared against register value **gcvSURF_SOURCE_MASK** —monochrome source mask defines transparency **gcvSURF_PATTERN_MASK** —pattern mask defines transparency

FgColor/BgColor	The values represent the foreground and background colors of the source. If the values are in destination format, set ColorConvert to **gcvFALSE**; otherwise, provide the values in A8R8G8B8 format and set ColorConvert to **gcvTRUE** to instruct the hardware to convert the values to the destination format before they are actually used.

gco2D_SetPixelMultiplyModesAdvanced

Description:
Sets the source and target pixel multiply modes. This function is compatible with Pixel Engine 2.0.

Syntax:
```
gceSTATUS
gco2D_SetPixelMultiplyModesAdvanced (
      IN gco2D                           Engine,
      IN gce2D_PIXEL_COLOR_MULTIPLY_MODE    SrcPremulitplySrcAlpha
      IN gce2D_PIXEL_COLOR_MULTIPLY_MODE    DstPremulitplyDstAlpha
      IN gce2D_GLOBAL_COLOR_MULTIPLY_MODE   SrcPremulitplyGlobalMode
      IN gce2D_PIXEL_COLOR_MULTIPLY_MODE    DstDemulitplyDstAlpha
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcPremultiplySrcAlpha	Source color premultiply with Source Alpha.
DstPremultiplyDstAlpha	Destination color premultiply with Destination Alpha.
SrcPremultiplyGlobalMode	Source color premulitply with Global color’s Alpha or Color.
DstDemultiplyDstAlpha	Destination color demultiply with Destination Alpha.

gco2D_SetPorterDuffBlending

Description:
Enables alpha blending engine in the hardware and sets the blending modes using the Porter-Duff defined blending rules.

Syntax:

```
gceSTATUS
gco2D_SetPorterDuffBlending (
      IN gco2D                   Engine,
      IN gce2D_PORTER_DUFF_RULE  Rule
);
```

Parameters:

Engine	Pointer to a gco2D object.
Rule	Porter-Duff blending rule.

gco2D_SetSource

Description:
Sets up the source rectangle.

Syntax:

```
gceSTATUS
gco2D_SetSource (
      IN gco2D          Engine,
      IN gcsRECT_PTR    SrcRect
);
```

Parameters:

Engine	Pointer to the gco2D object.
SrcRect	Pointer to a valid source rectangle.

gco2D_SetSourceColorKeyAdvanced

Description:
Sets the source color key. Color channel values should be specified in the range allowed by the source format. When the target format is A8, only Alpha components are used; otherwise, the Alpha components are not used. The function is compatible with PixelEngine 2.0 and above.

Syntax:

```
gceSTATUS
gco2D_SetSourceColorKeyAdvanced (
        IN gco2D            Engine,
        IN gctUINT32        ColorKey
);
```

Parameters:

Engine	Pointer to the gco2D object.
ColorKey	The color key value in A8R8G8B8 format.

gco2D_SetSourceColorKeyRangeAdvanced

Description:
Sets the source color key range. Color channel values should be specified in the range allowed by the source format. The lower color key's color channel values should be less than or equal to the corresponding color channel value of ColorKeyHigh. When the target format is A8, only Alpha components are used; otherwise, the Alpha components are not used. The function is compatible with PixelEngine 2.0 and above.

Syntax:

```
gceSTATUS
gco2D_SetSourceColorKeyRangeAdvanced (
        IN gco2D            Engine,
        IN gctUINT32        ColorKeyLow,
        IN gctUINT32        ColorKeyHigh
);
```

Parameters:

Engine	Pointer to the gco2D object.
ColorKeyLow	The low color key value in A8R8G8B8 format.
ColorKeyHigh	The high color key value in A8R8G8B8 format.

gco2D_SetSourceGlobalColorAdvanced

Description:
Sets the source global color value in A8R8G8B8 format. The function is compatible with PixelEngine 2.0 and above.

Syntax:

```
gceSTATUS
gco2D_SetSourceColorKeyAdvanced (
        IN gco2D            Engine,
        IN gctUINT32        Color32
);
```

Parameters:

Engine	Pointer to the gco2D object.
Color32	Source color in A8R8G8B8 format.

gco2D_SetStateU32

Description:
Sets 2D engine for 32 bit unsigned integer information.

Syntax:
```
gceSTATUS
gco2D_SetStateU32 (
IN gco2D Engine,
IN gce2D_STATE State,
IN gctUINT32 Value
);
```

Parameters:

Engine	Pointer to a gco2D object.
State	State to change.
Value	Value for the original state.

gco2D_SetStretchFactors

Description:
Calculates and programs the stretch factors.

Syntax:
```
gceSTATUS
gco2D_SetStretchFactors (
      IN gco2D         Engine,
      IN gctUINT32     HorFactor,
      IN gctUINT32     VerFactor
);
```

Parameters:

Engine	Pointer to a gco2D object.
HorFactor	Horizontal stretch factor.
VerFactor	Vertical stretch factor.

gco2D_SetStretchRectFactors

Description:
Calculates and programs the stretch factors based on rectangles.

Syntax:
```
gceSTATUS
gco2D_SetStretchRectFactors (
      IN gco2D           Engine,
      IN gcsRECT_PTR     SrcRect,
      IN gcsRECT_PTR     DestRect
);
```

Parameters:

Engine	Pointer to a gco2D object.
SrcRect	Pointer to a valid source rectangle.
DestRect	Pointer to a valid destination rectangle.

gco2D_SetTargetColorKeyAdvanced

Description:
Sets the target color key. Color channel values should be specified in the range allowed by the target format. When the target format is A8, only the Alpha component is used; otherwise, the Alpha component is not used. The function is compatible with PixelEngine 2.0 and above.

Syntax:
```
gceSTATUS
gco2D_SetTargetColorKeyAdvanced (
      IN gco2D           Engine,
      IN gctUINT32       ColorKey
);
```

Parameters:

Engine	Pointer to the gco2D object.
ColorKey	The color key value in A8R8G8B8 format.

gco2D_SetTargetColorKeyRangeAdvanced

Description:

Sets the target color key range. Color channel values should be specified in the range allowed by the target format. The lower color key's color channel values should be less than or equal to the corresponding color channel value of ColorKeyHigh. When the target format is A8, only Alpha components are used; otherwise, the Alpha components are not used. The function is compatible with PixelEngine 2.0 and above.

Syntax:

```
gceSTATUS
gco2D_SetTargetColorKeyRangeAdvanced (
      IN gco2D           Engine,
      IN gctUINT32       ColorKeyLow,
      IN gctUINT32       ColorKeyHigh
);
```

Parameters:

Engine	Pointer to the gco2D object.
ColorKeyLow	The low color key value in A8R8G8B8 format.
ColorKeyHigh	The high color key value in A8R8G8B8 format.

gco2D_SetTargetEx

Description:

Configures the destination.

Syntax:

Syntax:

```
gceSTATUS
gco2D_SetTarget (
      IN gco2D            Engine,
      IN gctUINT32        Address,
      IN gctUINT32        Stride,
      IN gceSURF_ROTATION Rotation,
      IN gctUINT32        SurfaceWidth,
      IN gctUINT32        SurfaceHeight
);
```

Parameters:

Engine	Pointer to a gco2D object.
Address	Destination surface base address.
Stride	Destination surface stride in bytes.
Rotation	If the destination surface is rotated 90 degrees, set to not zero.
SurfaceWidth	Destination surface width in pixels. Required only if the surface is rotated.
SurfaceHeight	Destination surface height in pixels. Required only if the surface is rotated in PixelEngine 2.0.

gco2D_SetTargetGlobalColorAdvanced

Description:
Sets the target global color value in A8R8G8B8. The function is compatible with PixelEngine 2.0 and above.

Syntax:

```
gceSTATUS
gco2D_SetTargetGlobalColorAdvanced (
      IN gco2D            Engine,
      IN gctUINT32        Color32
);
```

Parameters:

Engine	Pointer to the gco2D object.
Color32	Target color in A8R8G8B8 format.

gco2D_SetTransparencyAdvancedEx

Description:

Sets the source, target, and pattern transparency modes. In addition, enables or disables DFB color key mode.

This function is only working with full DFB 2D core.

Syntax:

```
gceSTATUS
gco2D_SetTransparencyAdvancedEx (
      IN gco2D                  Engine,
      IN gce2D_TRANSPARENCY     SrcTransparency,
      IN gce2D_TRANSPARENCY     DstTransparency,
      IN gce2D_TRANSPARENCY     PatTransparency,
      IN gctBOOL                EnableDFBColorKeyMode
);
```

Parameters:

Engine	Pointer to the gco2D object.
SrcTransparency	Source Transparency.
DstTransparency	Destination Transparency.
PatTransparency	Pattern Transparency.
EnableDFBColorKeyMode	Enable/disable DFB color key mode. The transparent pixels will be bypassed when enabling DFB color key mode; otherwise, those pixels may be processed by the following pipes.

gco2D_SetUserFilterKernel

Description:

Sets the filter kernel defined by user.

Syntax:

```
gceSTATUS
gco2D_SetUserFilterKernel (
      IN gco2D                 Engine,
      IN gceFILTER_PASS_TYPE   PassType,
      IN gctUINT16_PTR         KernelArray
);
```

Parameters:

Engine	Pointer to a gco2D object.
PassType	Pass type for the filter blit.
KernelArray	Pointer to the kernel array from user.

gco2D_StretchBlit

Description:
Stretch blit.

Syntax:

```
gceSTATUS
gco2D_StretchBlit (
      IN gco2D            Engine,
      IN gctUINT32        RectCount,
      IN gcsRECT_PTR      Rect,
      IN gctUINT8         FgRop,
      IN gctUINT8         BgRop,
      IN gceSURF_FORMAT   DestFormat
);
```

Parameters:

Engine	Pointer to a gco2D object.
RectCount	The number of rectangles to draw. The array of line positions, pointed to by the Position parameter, must have at least RectCount positions.
Rect	Points to an array of rectangles. All rectangles are assumed to be of the same size.
FgRop	Foreground ROP to use with the opaque pixels.
BgRop	Background ROP to use with the transparent pixels.
DestFormat	The format of the destination buffer.

gcoBRUSH_Destroy

Description:
Destroys a gcoBRUSH object.

Syntax:
```
gceSTATUS
gcoBRUSH_Destroy (
      IN gcoBRUSH         Brush
);
```

Parameters:

Brush	Pointer to a gcoBRUSH object that you want to destroy.

6 Surface Objects

gcoSURF_Blit

Description:
Generic rectangular blit.

Syntax:
```
gceSTATUS
gcoSURF_Blit (
      IN OPTIONAL gcoSURF                 SrcSurface,
      IN gcoSURF                          DestSurface,
      IN gctUINT32                        RectCount,
      IN OPTIONAL gcsRECT_PTR             SrcRect,
      IN gcsRECT_PTR                      DestRect,
      IN OPTIONAL gcoBRUSH                Brush,
      IN gctUINT8                         FgRop,
      IN gctUINT8                         BgRop,
      IN OPTIONAL gceSURF_TRANSPARENCY    Transparency,
      IN OPTIONAL gctUINT32               TransparencyColor,
      IN OPTIONAL gctPOINTER           Mask,
      IN OPTIONAL gceSURF_MONOPACK     MaskPack
  );
```

Parameters:

SrcSurface	Pointer to the surface source.
DestSurface	Pointer the surface destination.
RectCount	The number of rectangles to draw. The array of rectangles pointed to by Rect parameter must have at least RectCount items. Note that, for masked source blits, only one destination rectangle is supported.
SrcRect	If RectCount is 1, SrcRect represents an absolute rectangle within the source surface. If RectCount is greater than 1, right, bottom members of SrcRect are ignored. Left, top members are used as the offset from the origin of each destination rectangle in DestRect list to determine the corresponding source rectangle. In this case, the width and the height of the source are assumed the same as of the corresponding destination rectangle.
DestRect	Pointer to a list of destination rectangles.
Brush	The brush you want to use for the drawing.
FgRop	Foreground ROP to use with opaque pixels.
BgRop	Background ROP to use with transparent pixels.
Transparency	**gcvSURF_OPAQUE**—each pixel of the bitmap overwrites the destination. **gcvSURF_SOURCE_MATCH**—source pixels compared against register value to determine the transparency. In simple terms, the transparency comes down to selecting the ROP code to use. Opaque pixels use foreground ROP and transparent ones use background ROP. **gcvSURF_SOURCE_MASK**—monochrome source mask defines transparency. **gcvSURF_PATTERN_MASK**—pattern mask defines transparency.
TransparencyColor	This value is used in **gcvSURF_SOURCE_MATCH** transparency mode. The value is compared against each pixel to determine transparency of the pixel. If the values found equal, the pixel is transparent; otherwise, it is opaque.
Mask	A pointer to monochrome mask for masked source blits.
MaskPack	Determines how many horizontal pixels are there per each 32-bit chunk of monochrome mask. For example, if set to **gcvSURF_PACKED8**, each 32-bit chunk is 8-pixel wide. This also means that it defines 4 vertical lines of pixel mask.

gcoSURF_Construct

Description:

Creates a new gcoSURF object.

Syntax:

```
gceSTATUS
gcoSURF_Construct (
      IN gcoHAL           Hal,
      IN gctUINT          Width,
      IN gctUINT          Height,
      IN gctUINT          Depth,
      IN gceSURF_TYPE     Type,
      IN gceSURF_FORMAT   Format,
      IN gcePOOL          Pool,
      OUT gcoSURF *       Surface
);
```

Parameters:

Hal	Pointer to a gcoHAL object.
Width	The width, in pixels, of the surface you want to create.
Height	The height, in pixels, of the surface you want to create.
Depth	The depth, in pixels, of the surface you want to create.
Type	The surface type you want to create.
Format	The surface format you want to create.
Pool	The pool you want your new surface to allocate from.
Surface	Pointer to the variable that holds the gcoSURF object pointer.

gcoSURF_ConstructWrapper

Description:
Creates a new gcoSURF wrapper object.

Syntax:

```
gceSTATUS
gcoSURF_ConstructWrapper (
      IN gcoHAL            Hal,
      OUT gcoSURF *        Surface
);
```

Parameters:

Hal	Pointer to a gcoHAL object.
Surface	Pointer to the variable that will hold the gcoSURF object pointer.

gcoSURF_Destroy

Description:
Destroys a gcoSURF object.

Syntax:

```
gceSTATUS
gcoSURF_Destroy (
      IN gcoSURF           Surface
);
```

Parameters:

Surface	Pointer to a gcoSURF object that you want to destroy.

gcoSURF_DisableAlphaBlend

Description:
Disables the hardware alpha blending engine and engages the ROP engine. See also gcoSURF_EnableAlphaBlend.

Syntax:

```
gceSTATUS
gcoSURF_DisableAlphaBlend (
      IN gcoSURF           Surface
);
```

Parameters:

Surface	Pointer to the surface.

gcoSURF_EnableAlphaBlend

Description:

Disengages the ROP engine and enables the hardware alpha blending engine. See also gcoSURF_DiableAlphaBlend.

Syntax:

```
gceSTATUS
gcoSURF_EnableAlphaBlend (
      IN gcoSURF                      Surface,
      IN gctUINT8                     SrcGlobalAlphaValue,
      IN gctUINT8                     DstGlobalAlphaValue,
      IN gceSURF_PIXEL_ALPHA_MODE     SrcAlphaMode,
      IN gceSURF_PIXEL_ALPHA_MODE     DstAlphaMode,
      IN gceSURF_GLOBAL_ALPHA_MODE    SrcGlobalAlphaMode,
      IN gceSURF_GLOBAL_ALPHA_MODE    DstGlobalAlphaMode,
      IN gceSURF_BLEND_FACTOR_MODE    SrcFactorMode
      IN gceSURF_BLEND_FACTOR_MODE    DstFactorMode
      IN gceSURF_PIXEL_COLOR_MODE     SrcColorMode,
      IN gceSURF_PIXEL_COLOR_MODE     DstColorMode
);
```

Parameters:

Surface	Pointer to the surface.
SrcGlobalAlphaValue **DstGlobalAlphaValue**	Global alpha value, source, and destination for the color components.
SrcAlphaMode **DstAlphaMode**	The per-pixel, source, and destination of the alpha component mode.
SrcGlobalAlphaMode **DstGlobalAlphaMode**	The global per-pixel, source, and destination of the alpha value selection.
SrcFactorMode **DstFactorMode**	The final blending, source, and destination of the factor mode.
SrcColorMode **DstColorMode**	The per-pixel, source, and destination of the color mode.

gcoSURF_FilterBlit

Description:

Filter blit.

Syntax:

```
gceSTATUS
gcoSURF_FilterBlit (
      IN gcoSURF          SrcSurface,
      IN gcoSURF          DestSurface,
      IN gcsRECT_PTR      SrcRect,
      IN gcsRECT_PTR      DestRect,
      IN gcsRECT_PTR      DestSubRect
);
```

Parameters:

SrcSurface	Pointer to the source surface.
DestSurface	Pointer to the destination surface.
SrcRect	The coordinates of the entire source image.
DestRect	The coordinates of the entire destination image.
DestSubRect	The coordinates of a sub area to render within the destination. If DestSubRect is gcvNULL, the complete image is rendered using coordinates set by the destination image (DestRect). If DestSubRect is not gcvNULL, and DestSubRect and DestRect are not equal, DestSubRect is assumed to be within the destination image (DestRect) and is used to render the sub area only.

gcoSURF_Flush

Description:
Flushes the caches to ensure that the surface has all pixels.

Syntax:

```
gceSTATUS
```

```
gcoSURF_Flush (
      IN gcoSURF          Surface
);
```

Parameters:

Surface	Pointer to the surface.

gcoSURF_GetAlignedSize

Description:

Gets the aligned size of a gcoSURF object.

Syntax:

```
gceSTATUS
gcoSURF_GetAlignedSize (
      IN gcoSURF          Surface,
      OUT gctUINT *       Width,
      OUT gctUINT *       Height,
      OUT gctINT *        Stride
);
```

Parameters:

Surface	Pointer to a gcoSURF object.
Width	Pointer to a variable that receives the aligned width of the gcoSURF object. If ‘Width’ is **gcvNULL**, no width information is returned.
Height	Pointer to a variable that receives the aligned height of the gcoSURF object. If ‘Height’ is **gcvNULL**, no height information is returned.
Stride	Pointer to a variable that receives the stride of the gcoSURF object. If ‘Stride’ is **gcvNULL**, no stride information is returned.

gcoSURF_GetFormat

Description:

Gets the surface type and format.

Syntax:

```
gceSTATUS
gcoSURF_GetFormat (
      IN gcoSURF             Surface
      OUT gceSURF_TYPE *     Type,
        OUT gceSURF_FORMAT *   Format
);
```

Parameters:

Surface	Pointer to a gcoSURF object.
Type	Pointer to a variable that receives the type of the gcoSURF object. If 'Type' is **gcvNULL**, no type information is returned.
Format	Pointer to a variable that receives the format of the gcoSURF object. If 'Format' is **gcvNULL**, no format information is returned.

gcoSURF_GetSize

Description:
Gets the size of a gcoSURF object.

Syntax:

```
gceSTATUS
gcoSURF_GetSize (
        IN gcoSURF          Surface,
        OUT gctUINT *       Width,
        OUT gctUINT *       Height,
        OUT gctUINT *       Depth
);
```

Parameters:

Surface	Pointer to a gcoSURF object.
Width	Pointer to a variable that receives the width of the gcoSURF object. If 'Width' is **gcvNULL**, no width information is returned.
Height	Pointer to a variable that receives the height of the gcoSURF object. If 'Height' is **gcvNULL**, no height information is returned.
Depth	Pointer to a variable that receives the depth of the gcoSURF object. If 'Depth' is **gcvNULL**, no depth information is returned.

gcoSURF_Line

Description:
Draws one or more Bresenham lines.

Syntax:

```
gceSTATUS
gcoSURF_Line (
        IN gcoSURF          Surface,
        IN gctUINT32        LineCount,
        IN gcsRECT_PTR      Position,
        IN gcoBRUSH         Brush,
        IN gctUINT8         FgRop,
        IN gctUINT8         BgRop
    );
```

Parameters:

Surface	Pointer to a gcoSURF object.
LineCount	The number of lines you want drawn. The array of line positions pointed to by the Position parameter requires at least the LineCount item.
Position	Points to an array of positions in (x0, y0) - (x1, y1) format.
Brush	The brush to use for drawing the lines.
FgRop	Foreground ROP to use with opaque pixels.
BgROP	Background ROP to use with transparent pixels.

gcoSURF_Lock

Description:

Locks the surface.

Syntax:

```
    gceSTATUS
    gcoSURF_Lock (
        IN gcoSURF                  Surface,
        OPTIONAL OUT gctUINT32 *    Address,
        OPTIONAL OUT gctPOINTER *   Memory
    );
```

Parameters:

Surface	Pointer to a gcoSURF object.

Address	Physical address array of the surface: For YV12, Address[0] is for Y channel, Address[1] is for V channel and Address[2] is for U channel; For I420, Address[0] is for Y channel, Address[1] is for U channel and Address[2] is for V channel; For NV12, Address[0] is for Y channel and Address[1] is for UV channel; For all other formats, only Address[0] is used to return the physical address.
Memory	Logical address array of the surface: For YV12, Memory[0] is for Y channel, Memory[1] is for V channel and Memory[2] is for U channel; For I420, Memory[0] is for Y channel, Memory[1] is for U channel and Memory[2] is for V channel; For NV12, Memory[0] is for Y channel and Memory[1] is for UV channel; For all other formats, only Memory[0] is used to return the logical address.

gcoSURF_MonoBlit

Description:

Monochrome blit.

Syntax:

```
gceSTATUS
gcoSURF_MonoBlit (
      IN gcoSURF                  DestSurface,
      IN gctPOINTER               Source,
      IN gceSURF_MONOPACK         SourcePack,
      IN gcsPOINT_PTR             SourceSize,
      IN gcsPOINT_PTR             SourceOrigin,
      IN gcsRECT_PTR              DestRect,
      IN OPTIONAL gcoBRUSH        Brush,
      IN gctUINT8                 FgRop,
      IN gctUINT8                 BgRop,
      IN gctBOOL                  ColorConvert,
      IN gctUINT8                 MonoTransparency
```

```
        IN gceSURF_TRANSPARENCY   Transparency,
        IN gctUINT32              FgColor,
        IN gctUINT32              BgColor,
    );
```

Parameters:

DestSurface	Pointer to the destination surface.
Source	A pointer to the monochrome bitmap.
SourcePack	Determines how many horizontal pixels there are per each 32-bit chunk of monochrome bitmap. For example, if set to gcvSURF_PACKED8, each 32-bit chunk is 8-pixel wide. This also means that it defines 4 vertical lines of pixels.
SourceSize	Size of the source monochrome bitmap in pixels.
SourceOrigin	Top left coordinate of the source within the bitmap.
DestRect	Pointer to a list of destination rectangles.
Brush	Brush you want to use for drawing.
FgRop	Foreground ROP to use with opaque pixels.
BgRop	Background ROP to use with transparent pixels.
ColorConvert	The values of FgColor and BgColor parameters are stored directly in internal color registers and are used either directly as the source color, or converted to the destination format before actually used. The later happens if ColorConvert is not zero.
MonoTransparency	This value is used in **gcvSURF_SOURCE_MATCH** transparency mode. The value can be either 0 or 1 and is compared against each mono pixel to determine transparency of the pixel. If a match is found for a pixel, the pixel is transparent; otherwise, it is opaque.
Transparency	**gcvSURF_OPAQUE**—each pixel of the bitmap overwrites the destination. **gcvSURF_SOURCE_MATCH**—source pixels compared against register value to determine the transparency. In simple terms, the transparency comes down to selecting the ROP code to use. Opaque pixels use foreground ROP. Transparent ones use background ROP. **gcvSURF_SOURCE_MASK**—monochrome source mask defines transparency. **gcvSURF_PATTERN_MASK**—pattern mask defines transparency.
FgColor/BgColor	The values are used to represent foreground and background colors of the source. If the values are in destination format, set ColorConvert to 0; otherwise, provide the values in ARGB8 format and set ColorConvert to 1 to instruct the hardware to convert the

values to the destination format before they are actually used.

gcoSURF_SetBuffer

Description:
Sets the underlying buffer for the surface wrapper.

Syntax:

```
gceSTATUS
gcoSURF_SetBuffer (
      IN gcoSURF          Surface
      IN gceSURF_TYPE     Type,
      IN gceSURF_FORMAT   Format,
      IN gctUINT          Stride,
      IN gctPOINTER       Logical,
      IN gctUINT32        Physical
);
```

Parameters:

Surface	Pointer to the gcoSURF object.
Type	Type of surface to create.
Format	Format of surface to create.
Stride	Surface stride. If set to ~0, the stride will be auto-computed.
Logical	Logical pointer to the user allocated surface or **gcvNULL** if no logical pointer has been provided.
Physical	Physical address.

gcoSURF_SetClipping

Description:
Sets clipping rectangle.

Syntax:

```
gceSTATUS
gcoSURF_SetClipping (
      IN gcoSURF          Surface
);
```

Parameters:

Surface	Pointer to a gcoSURF object.

gcoSURF_SetDither

Description:
Sets the surface dither flag.

Syntax:

```
gceSTATUS
gcoSURF_SetDither (
      IN gcoSURF            Surface
      IN gctBOOL            Dither
);
```

Parameters:

Surface	Pointer to a gcoSURF object.
Dither	Ditherable or not.

gcoSURF_SetWindow

Description:
Sets the size of the surface in pixels, and if necessary, maps the underlying buffer set by gcoSURF_SetBuffer.

Syntax:

```
gceSTATUS
gcoSURF_SetWindow (
      IN gcoSURF          Surface,
      IN gctUINT          X,
      IN gctUINT          Y,
      IN gctUINT          Width,
      IN gctUINT          Height
);
```

Parameters:

Surface	Pointer to the surface.
X and Y	The X and Y origin coordinates of the surface.
Width	The width of the surface in pixels.
Height	The height of the surface in pixels.

gcoSURF_Unlock

Description:
Unlocks the surface.

Syntax:

```
gceSTATUS
gcoSURF_Unlock (
        IN gcoSURF          Format,
        IN gctPOINTER       Memory
);
```

Parameters:

Surface	Pointer to a gcoSURF object.
Memory	Pointer to mapped memory.

7 Rectangle Objects

gcsRECT_Height

Description:
Returns the height of the rectangle.

Syntax:

```
gceSTATUS
gcsRECT_Height (
        IN gcsRECT_PTR       Rect,
        OUT gctINT32 *       Height
);
```

Parameters:

Rect	Pointer to a valid rectangle structure.
Height	Pointer to a variable that receives the height of the rectangle.

gcsRECT_IsEqual

Description:
Compares two rectangles. See also **gcsRECT_IsOfEqualSize**.

Syntax:

```
gceSTATUS
gcsRECT_IsEqual (
      IN gcsRECT_PTR      Rect1,
      IN gcsRECT_PTR      Rect2,
      OUT gctBOOL *       Equal
);
```

Parameters:

Rect1	Pointer to a valid rectangle structure to compare.
Rect2	Pointer to a valid rectangle structure to compare.
Equal	Pointer to a variable that receives a **gcvTRUE** if the rectangles are equal. Returns a **gcvFALSE** if the rectangles are not equal.

gcsRECT_IsOfEqualSize

Description:

Compares the sizes of two rectangles. See also **gcsRECT_IsEqual**.

Syntax:

```
gceSTATUS
gcsRECT_IsOfEqualSize (
      IN gcsRECT_PTR      Rect1,
      IN gcsRECT_PTR      Rect2,
      OUT gctBOOL *       EqualSize
);
```

Parameters:

Rect1	Pointer to a valid rectangle structure.
Rect2	Pointer to a valid rectangle structure to compare.
EqualSize	Pointer to a variable that receives a **gcvTRUE** if the rectangles are of equal size. Returns a **gcvFALSE** if the rectangles are not of equal size.

gcsRECT_Normalize

Description:

Ensures that the top left corner is at the left and is above the right bottom.

Syntax:

```
gceSTATUS
```

```
gcsRECT_Normalize (
      IN OUT gcsRECT_PTR  Rect,
);
```

Parameters:

Rect	**IN**: Pointer to a valid rectangle structure. **OUT**: Normalized rectangle

gcsRECT_Rotate

Description:

Computes the related rotation based on orientation.

Syntax:

```
gceSTATUS
gcsRECT_Rotate (
      IN OUT gcsRECT_PTR    Rect,
      IN gceSURF_ROTATION   Rotation,
      IN gceSURF_ROTATION   toRotation,
      IN gctINT32           SurfaceWidth,
      IN gctINT32           SurfaceHeight
);
```

Parameters:

Rect	**IN**: Pointer to the rectangle to be rotated. **OUT**: Pointer to the rectangle which has been rotated to toRotation.
Rotation	Original rotation.
toRotation	Target rotation.
SurfaceWidth	The width of the surface.
SurfaceHeight	The height of the surface.

gcsRECT_Set

Description:
Initializes a rectangle structure.

Syntax:
```
gceSTATUS
gcsRECT_Set (
      OUT gcsRECT_PTR     Rect,
      IN gctINT32         Left,
      IN gctINT32         Top,
      IN gctINT32         Right,
      IN gctINT32         Bottom
);
```

Parameters:

Rect	Initialize a rectangle structure.
Left	Set the left coordinates of the rectangle.
Top	Set the top coordinates of the rectangle.
Right	Set the right coordinates of the rectangle.
Bottom	Set the bottom coordinates of the rectangle.

gcsRECT_Width

Description:
Returns the width of the rectangle.

Syntax:
```
gceSTATUS
gcsRECT_Width (
      IN gcsRECT_PTR      Rect,
      OUT gctINT32 *      Width
);
```

Parameters:

Rect	Pointer to a valid rectangle structure.
Width	Pointer to a variable that receives the width of the rectangle.

8 Basic 2D Operations

Some operations described in this section require feature support in the GPU hardware. If any operation described in this section is inconsistent with the processor reference manual, the reference manual takes precedence.

8.1 Line

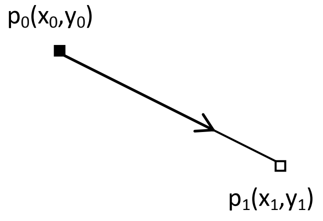
The LINE operation draws a line. Coordinates for two points are given as start point and end point. The end point is not drawn.

Lines are rendered using the Bresenham algorithm.

The Bresenham algorithm has the advantage of using integer arithmetic and has no accumulation of rounding errors.

In the case of a line, only ROP2 and ROP4 are supported. It operates on pattern and destination. The pattern should have a transparency mask in order to use ROP4.

Clipping is supported for lines on a per pixel basis.



8.2 Rectangle Fill and Clear

Rectangle fill suffuses a rectangle area with a given color. Essentially, rectangle fill is a pattern fill where an 8x8 pattern is initialized with the specified color. It supports ROP2 and ROP4 with the pattern and destination as its inputs. If ROP4 is used, the pattern should have a transparency mask.

Clear is similar to rectangle fill except that it does not does not use a pattern. A 32-bit clear value with 4-bit byte mask is used to fill the entire rectangle area.

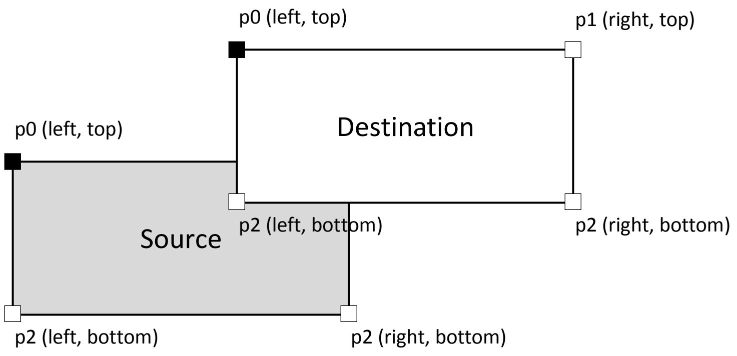
Both rectangle fill and clear support clipping, which is performed on a per primitive basis.

8.3 Bit BLT

Bit blit transfers data from one area of memory (source) to another area of memory (destination). The source and destination can be from the same or different memory locations. Both source and destination must be described by a rectangular area. The source and destination rectangles can be of the same size (most bit blits are of this nature), or they can be of different sizes in which case the operation becomes a stretch or shrink blit.

Bit blit supports ROP2, ROP3, and ROP4 which includes source, destination and pattern, and an optional transparency color.

Clipping can be performed on a primitive basis.



The Bit BLT primitive supports the following 10 source and 7 destination image formats:

Formats	Source Image	Destination Image
A1R5G5B5	Yes	Yes
A4R4G4B4	Yes	Yes
X1R5G5B5	Yes	Yes
X4R4G4B4	Yes	Yes
R5G6B5	Yes	Yes
A8R8G8B8	Yes	Yes
X8R8G8B8	Yes	Yes
A8	Yes	No
1-bit monochrome	Yes	No
8-bit color index	Yes	No

8.4 Stretch BLT

The Stretch BLT primitive performs a Bit BLT operation with stretch or shrink. The modified Bresenham algorithm is used to generate corresponding coordinates for fast stretching. The stretch factor is specified in a 15.0 fixed-point format. Stretch blit is not allowed to overlap. That is, no part of source and destination can share any piece of memory. Non-stretch blits can overlap. Stretch blit clipping is performed on a per pixel basis.

8.5 Monochrome Expansion and Mask BLT

Monochrome expansion and mask blit are different operations even though both use the bit stream from command buffer. Both can be the source for ROP4 source selection. This means that each output pixel can be a combination of source, pattern, monochrome mask (for masked blits), and destination.

Monochrome Expansion

For monochrome expansion, the bit from the stream is used to switch on/off a solid color that is defined in a register. This mechanism enables the use of just one bit per pixel to represent colors. In effect, the monochrome expansion primitive increases color representation from one bit per pixel to multiple bits per pixel. A typical application for mono color is font drawing.
Monochrome expansion does not support overlapping of the source and destination. It is the responsibility of the driver to ensure that the command will never be executed on overlapping source and destination.

Mask BLT
For Mask BLT, the bit from the stream is used to toggle on/off a color in the source frame buffer. Mask BLT takes its color source from memory and its monochrome mask from the command stream. Clipping is supported and is performed on a per pixel basis.

8.6 Filter BLT

Filter blit performs high quality scaling, up or down, using an FIR re-sampling filter with up to 9 taps. Sub-pixel coordinates (locations between the pixel grids) are generated by the drawing engine. The filter block in the drawing engine uses the sub-pixel information to select the appropriate filter kernel. All i.MX 6 processors process 1 pixel every cycle when performing filter blit.

A stretch- or shrink-factor of 15.16 fixed-point format is supported. To generate a single destination pixel requires 9 source pixels. An image is scaled in two passes: one for X-dimension (HOR_FILTER_BLT), and the other for Y-dimension (VER_FILTER_BLT). Software sets up the filter kernel/coefficient table and the kernel size as well as a temporary buffer for storing intermediate results. After the first pass is completed, intermediate results are sent back to memory. Then, the second pass starts to scale the first-pass image. Because of this two-step procedure, the throughput of Filter BLT is lower than that of Stretch BLT. In addition, the Filter Kernel Table may need to be reloaded and some cycles are consumed in calculating the stepping parameters.

When the stretch or shrink factor is 1, the filter blit works as a bit blit copy. It can be used as format converter, for example, YUV to RGB converter. To use as a format converter, only one pass (HOR_FILTER_BLT or VER_FILTER_BLT) is needed. To optimize the memory bandwidth when using filter blit to do YUV to RGB filtering, the temporary target buffer format can be specified as YUY2 to process Y-dimension filtering (VER_FILTER_BLT). This is to avoid converting YUV to A8R8G8B8 in the 1st vertical pass to reduce the memory bandwidth and increase the pixel processing rate. This is the only special case that GPU may use YUY2 as target format.

The Filter BLT primitive supports the following 13 source and 7 destination image formats:

Formats		Source Image	Destination Image
A1R5G5B5		Yes	Yes
A4R4G4B4		Yes	Yes
A8R8G8B8		Yes	Yes
R5G6B5		Yes	Yes
X1R5G5B5		Yes	Yes
X4R4G4B4		Yes	Yes
X8R8G8B8		Yes	Yes
YUV	NV12 (4:2:0, 2 planes)	Yes	No
	NV16 (4:2:2, 2 planes)	Yes	No
	UYVY (4:2:2, interleave)	Yes	No
	YUY2 (4:2:2, interleave)	Yes	No
	YV12 (4:2:0, 3 planes)	Yes	No
	8-bit color index	Yes	No

Filter blit summary:

- Color space conversion between YUY2 and RGB.
- High quality re-sampling filter with kernel sizes of 1, 3, 5, 7, and 9.
- Stretch factor of format 15.16 fixed-point is supported.

- Programmable filter coefficients.
- Filter blit supports alpha blending.
- Filter blit supports rotation.
- Filter blit supports bandwidth reduction between vertical and horizontal scaling.
- Clipping is supported and is performed on per pixel basis.

8.7 Other Operations

8.7.1 ROP Support

ROP2 supports 16 ROP types. ROP3 and ROP4 support 256 ROP types.

8.7.2 Rotation

90° / 180° / 270° / X-Flip / Y-Flip / Mirror rotation is supported for all primitives.

8.7.3 Transparency Mode

For monochrome expansion:

- Opaque
- Conditional transparency. Transparent if the current pixel matches the specified value.

For blits:

- Opaque
- Masked transparency. Transparent if the mask for the current pixel or pattern is zero.
- Source Conditional transparency. Transparent if the source pixel is within the specified value range.
- Destination Conditional transparency. Transparent if the destination pixel is not within the specified value range.

8.7.4 Clipping

One clipping rectangle is supported for all bit blit primitives.

8.7.5 Data Formats

The graphics engine supports 14 source data formats. In addition to these 14 source formats, for RGB source formats, the GPU also supports swizzle formats (ARGB, RGBA, ABGR, BGRA). For YUV formats, GPU supports their U/V swap formats.

- A1R5G5B5
- A4R4G4B4
- A8R8G8B8
- R5G6B5
- X1R5G5B5
- X4R4G4B4
- X8R8G8B8
- A8
- NV12
- NV16
- UYVY (4:2:2)

- YUY2 (4:2:2)
- YV12 (4:2:0)
- 8-bit color index

There are 7 destination data formats supported by the graphics engine. In addition to these destination RGB formats, swizzle formats (ARGB, RGBA, ABGR, BGRA) are also supported.

- A1R5G5B5
- A4R4G4B4
- A8R8G8B8
- R5G6B5
- X1R5G5B5
- X4R4G4B4
- X8R8G8B8

8.7.6 ARGB Data Conversion

The pixels read from source or destination will be expanded into A8R8G8B8 format to maintain lossless pixel operations. The resulting pixels will be converted into the destination format.

8.7.7 YUV to RGB Conversion

YUV data can be converted into 8-bit per component RGB format at the output of the cache only. Once YUV data is converted to RGB format, conversion back to YUV format is not possible. The GPU supports BT.601 YUV to RGB color conversion standards.
The YUV to RGB conversion is done using the following approximation:

$16 \leq Y \leq 235$
$16 \leq U \leq 240$
$16 \leq V \leq 240$

9 Revision History

This section describes top level differences between document revisions:

Version	Date	Driver Version	Notes
1.1	2013-01-09	4.6.9-p8	Added new API, gco2D_SetStateU32().
1.0	2012-10-04	4.6.9-p6	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support