

GSR-003
rev 1.10.00***EGL 1.0***

변경 내역

버전	배포 날짜	변경사항
1.10.00	2005.10.11	초안작성

지적 재산권에 대한 공지

본 문서 및 문서와 함께 제공되는 일체의 정보는 SK 텔레콤의 WIPI 추가 규격을 명시함으로써, SK 텔레콤에 납품하는 단말기 제조사들 및 콘텐츠 개발업체가 WIPI 추가 규격을 구현하도록 하기 위함이다. 본 문서 및 문서와 함께 제공되는 일체의 정보는 SK 텔레콤의 소유물이다. 이의 일부 또는 전부는 SKT WIPI 를 위해서만 SKT 의 승인 아래 복사되거나 배포될 수 있다.

에스케이텔레콤 주식회사

목 차

1 일반사항	6
1.1 개요	6
1.2 목적	6
1.3 규격의 범위.....	6
1.4 용어정의	6
2 의존성	7
2.1 WIPI 의존성.....	7
2.2 GIGA 의존성	7
2.3 GIGA 의존성	7
2.4 OAT 의존성.....	7
3 제조사 포팅 가이드	8
4 개발자 가이드	9
4.1 Type 정의.....	9
4.2 C API	9

세부목차

1 일반사항	6
1.1 개요	6
1.2 목적	6
1.3 규격의 범위.....	6
1.4 용어정의	6
2 의존성	7
2.1 WIPI 의존성.....	7
2.2 GIGA 의존성	7
2.3 GIGA 의존성	7
2.4 OAT 의존성.....	7
3 제조사 포팅 가이드	8
4 개발자 가이드	9
4.1 Type 정의.....	9
4.2 C API	9
4.2.1 OEMC_eglGetError.....	12
4.2.2 OEMC_eglGetDisplay.....	14
4.2.3 OEMC_eglInitialize	15
4.2.4 OEMC_eglTerminate	16
4.2.5 OEMC_eglQueryString	17
4.2.6 OEMC_eglGetConfigs	18
4.2.7 OEMC_eglChooseConfig	20
4.2.8 OEMC_eglGetConfigAttrib.....	25
4.2.9 OEMC_eglCreateWindowSurface	29
4.2.10 OEMC_eglCreatePixmapSurface	31
4.2.11 OEMC_eglCreatePbufferSurface.....	33
4.2.12 OEMC_eglDestroySurface	35
4.2.13 OEMC_eglQuerySurface	36
4.2.14 OEMC_eglCreateContext	38
4.2.15 OEMC_eglDestroyContext	40
4.2.16 OEMC_eglMakeCurrent	41
4.2.17 OEMC_eglGetCurrentContext	43
4.2.18 OEMC_eglGetCurrentSurface	44
4.2.19 OEMC_eglGetCurrentDisplay.....	45

4.2.20 OEMC_eglQueryContext	46
4.2.21 OEMC_eglWaitGL	48
4.2.22 OEMC_eglWaitNative	49
4.2.23 OEMC_eglSwapBuffers	50
4.2.24 OEMC_eglCopyBuffers	51
4.2.25 OEMC_eglGetProcAddress	52

1 일반사항

1.1 개요

This chapter explains Embedded GL specification v1.0 API part of GIGA 3D. All API description is follow Embedded GL specification v1.0, Additionally, GIGA 3D define minimum requirement about rendering target format. This requirement will be explained in function description..

EGL is designed to usable on multiple operating system and native window system. Rendering target setting and LCD interfacing is unified by EGL interface. User can query available device configuration list from EGL and choose one which device configuration will use.

1.2 목적

본 규격은 SK Telecom(주)(이하 “SK Telecom” 이라 한다.)의 EGL 1.0 을 위한 요구 사항 및 규격을 명시하는데 목적이 있다.

1.3 규격의 범위

본 문서는 SK Telecom에서 서비스 예정인 EGL 1.0을 위한 규격에 대하여 기술한다.

1.4 용어정의

본 문서에서 언급되는 용어들을 정의한다.

2 의존성

본 규격 구현을 위하여, 필요로 하는 WIPI 버전과, 타 WSR 및 본 규격의 검증을 위해 필요한 OAT 버전을 명시한다.

2.1 WIPI 의존성

본 WSR 규격은 아래 WIPI Core 버전 이상이 적용된 단말에 포팅 되어야 한다.

Base WIPI Core 버전
WIPI 1.08.08

2.2 GIGA 의존성

본 GSR 규격은 아래 GIGA Core 버전 이상이 적용된 단말에 포팅 되어야 한다.

Base GIGA Core 버전
GIGA 2.00.00

2.3 GIGA 의존성

본 GSR을 포팅하기 위하여서는 아래 명시된 GSR들을 모두 포팅 하여야 한다.

의존성을 갖는 GSR	설 명
GSR-001	OpenGL ES 1.0 common lite

2.4 OAT 의존성

본 규격의 테스트를 위해서는 다음 버전의 OAT가 필요하다.

3 제조사 포팅 가이드

(*) 추후 배포 예정임

4 개발자 가이드

4.1 Type 정의

Type	Description
M_EGLint	int
M_EGLBoolean	unsigned int
M_EGLConfig	void*
M_EGLContext	void*
M_EGLDisplay	void*
M_EGLSurface	void*
M_NativeDisplayType	M_EGLint
M_NativeWindowType	void*
M_NativePixmapType	void*

4.2 C API

Function	Description
M_EGLint OEMC_eglGetError (void);	The OEMC_eglGetError function returns error information of EGL
M_EGLDisplay OEMC_eglGetDisplay (M_NativeDisplayType displayID)	Getting EGLDisplay of specified native display
M_EGLBoolean OEMC_eglInitialize (M_EGLDisplay dpy, M_EGLint* major, M_EGLint* minor)	Initialize EGL interface system
M_EGLBoolean OEMC_eglTerminate (M_EGLDisplay dpy)	Terminate EGL interface system
const char* OEMC_eglQueryString (M_EGLDisplay dpy, M_EGLint name)	Returning EGL version and vendor string
M_EGLBoolean OEMC_eglGetConfigs (M_EGLDisplay dpy, M_EGLConfig* configs, M_EGLint config_size, M_EGLint* num_config)	Query EGL configuration list
M_EGLBoolean OEMC_eglChooseConfig (M_EGLDisplay dpy, const M_EGLint*	Selective query of EGL configuration list

attrib_list, M_EGLConfig* configs, M_EGLint config_size, M_EGLint* num_config)	
M_EGLBoolean OEMC_eglGetConfigAttrib (M_EGLDisplay dpy, M_EGLConfig config, M_EGLint attribute, M_EGLint* value)	Query EGL configuration attribute value
M_EGLSurface OEMC_eglCreateWindowSurface (M_EGLDisplay dpy, M_EGLConfig config, M_NativeWindowType win, const M_EGLint* attrib_list)	Create Native Window Surface
M_EGLSurface OEMC_eglCreatePixmapSurface (M_EGLDisplay dpy, M_EGLConfig config, M_NativePixmapType pixmap, const M_EGLint* attrib_list)	Create Native Pixmap Surface
M_EGLSurface OEMC_eglCreatePbufferSurface (M_EGLDisplay dpy, M_EGLConfig config, const M_EGLint* attrib_list)	Create pBuff surface of OES system
M_EGLBoolean OEMC_eglDestroySurface (M_EGLDisplay dpy, M_EGLSurface surface)	Destory surface
M_EGLBoolean OEMC_eglQuerySurface (M_EGLDisplay dpy, M_EGLSurface surface, M_EGLint attribute, M_EGLint* value)	Query attribute of surface
M_EGLContext OEMC_eglCreateContext (M_EGLDisplay dpy, M_EGLConfig config, M_EGLContext share_list, const M_EGLint* attrib_list)	Create context

M_EGLBoolean OEMC_eglDestroyContext (M_EGLDisplay dpy, M_EGLContext ctx)	Destroy context
M_EGLBoolean OEMC_eglMakeCurrent (M_EGLDisplay dpy, M_EGLSurface adraw, M_EGLSurface aread, M_EGLContext ctx)	Make current context
M_EGLSurface OEMC_eglGetCurrentSurface (M_EGLint readdraw)	Query currently selected context
M_EGLSurface OEMC_eglGetCurrentSurface (M_EGLint readdraw)	Query currently linked surface
M_EGLDisplay OEMC_eglGetCurrentDisplay (void)	Query currently linked display
M_EGLBoolean OEMC_eglQueryContext (M_EGLDisplay dpy, M_EGLContext ctx, M_EGLint attribute, M_EGLint* value)	Query attribute of context
M_EGLBoolean OEMC_eglWaitGL (void)	Wait until OES operation is done
M_EGLBoolean OEMC_eglWaitNative (M_EGLint engine)	Wait until Native system operation is done
M_EGLBoolean OEMC_eglSwapBuffers (M_EGLDisplay dpy, M_EGLSurface surface)	Swap buffer contents
M_EGLBoolean OEMC_eglCopyBuffers (M_EGLDisplay dpy, M_EGLSurface surface, M_NativePixmapType target)	Copy surface contents to target pixmap
void (*OEMC_eglGetProcAddress(const char *procname))	Return a GL or an EGL extension function

4.2.1 OEMC_eglGetError

설명

The OEMC_eglGetError function returns error information of EGL.

프로토타입

M_EGLint OEMC_eglGetError (void)

매개변수

없음

참고 사항

glGetError returns the value of the error flag. Each detectable error is assigned a numeric code and symbolic name. When an error occurs, the error flag is set to the appropriate error code value. No other errors are recorded until glGetError is called, the error code is returned, and the flag is reset to GL_NO_ERROR. If a call to glGetError returns GL_NO_ERROR, there has been no detectable error since the last call to glGetError, or since the GL was initialized.

To allow for distributed implementations, there may be several error flags. If any single error flag has recorded an error, the value of that flag is returned and that flag is reset to GL_NO_ERROR when glGetError is called. If more than one flag has recorded an error, glGetError returns and clears an arbitrary error flag value. Thus, glGetError should always be called in a loop, until it returns GL_NO_ERROR, if all error flags are to be reset.

Initially, all error flags are set to GL_NO_ERROR.

The following errors are currently defined:

GL_NO_ERROR

No error has been recorded. The value of this symbolic constant is guaranteed to be 0.

GL_INVALID_ENUM

An unacceptable value is specified for an enumerated argument. The offending command is ignored, and has no other side effect than to set the error flag.

GL_INVALID_VALUE

A numeric argument is out of range. The offending command is ignored, and has no other side effect than to set the error flag.

GL_INVALID_OPERATION

The specified operation is not allowed in the current state. The offending command is ignored, and has no other side effect than to set the error flag.

GL_STACK_OVERFLOW

This command would cause a stack overflow. The offending command is ignored, and has no other side effect than to set the error flag.

GL_STACK_UNDERFLOW

This command would cause a stack underflow. The offending command is ignored, and has no other side effect than to set the error flag.

GL_OUT_OF_MEMORY

There is not enough memory left to execute the command. The state of the GL is undefined, except for the state of the error flags, after this error is recorded.

When an error flag is set, results of a GL operation are undefined only if **GL_OUT_OF_MEMORY** has occurred. In all other cases, the command generating the error is ignored and has no effect on the GL state or frame buffer contents. If the generating command returns a value, it returns 0. If `glGetError` itself generates an error, it returns 0.

4.2.2 OEMC_eglGetDisplay

설명

Getting EGLDisplay of specified native display

프로토타입

M_EGLDisplay OEMC_eglGetDisplay (M_NativeDisplayType displayID)

매개변수

displayID

[in]

Specifies the display to connect to. EGL_DEFAULT_DISPLAY indicates the default display. EGL_DEFAULT_DISPLAY is same effect with G2D_LCD_DEFAULT and other LCD indicating IDs are follow GIGA 2D LCD type definition.

참고사항

eglGetDisplay obtains the EGL display connection for the native display native_display.

If display_id is EGL_DEFAULT_DISPLAY, a default display connection is returned.

If no display connection matching native_display is available, EGL_NO_DISPLAY is returned. No error is generated.

Use eglInitialize to initialize the display connection.

4.2.3 OEMC_eglInitialize

설명

Initialize EGL interface system

프로토타입

```
M_EGLBoolean OEMC_eglInitialize ( M_EGLDisplay dpy, M_EGLint* major,  
M_EGLint* minor )
```

매개변수

dpy

Specifies the EGL display connection to initialize.

major

Returns the major version number of the EGL implementation. May be NULL.

minor

Returns the minor version number of the EGL implementation. May be NULL.

참고사항

eglInitialize initialized the EGL display connection obtained with eglGetDisplay. Initializing an already initialized EGL display connection has no effect besides returning the version numbers.

major and minor do not return values if they are specified as NULL.

Use eglTerminate to release resources associated with an EGL display connection.

에러 코드

EGL_FALSE is returned if eglInitialize fails, EGL_TRUE otherwise. major and minor are not modified when EGL_FALSE is returned.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display cannot be initialized.

4.2.4 OEMC_eglTerminate

설명

Terminate EGL interface system

프로토타입

M_EGLBoolean OEMC_eglTerminate (M_EGLDisplay dpy)

매개변수

dpy

Specifies the EGL display connection to terminate.

참고사항

eglTerminate releases resources associated with an EGL display connection. Termination marks all EGL resources associated with the EGL display connection for deletion. If contexts or surfaces associated with display is current to any thread, they are not released until they are no longer current as a result of eglMakeCurrent.

Terminating an already terminated EGL display connection has no effect. A terminated display may be re-initialized by calling eglInitialize again.

에러 코드

EGL_FALSE is returned if eglTerminate fails, EGL_TRUE otherwise.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

4.2.5 OEMC_eglQueryString

설명

Returning EGL version and vendor string

프로토타입

```
const char* OEMC_eglQueryString          ( M_EGLDisplay dpy, M_EGLint name )
```

매개변수

dpy

Specifies the EGL display connection.

name

Specifies a symbolic constant, one of EGL_VENDOR, EGL_VERSION, or EGL_EXTENSIONS.

참고사항

eglQueryString returns a pointer to a static string describing an EGL display connection.
name can be one of the following:

EGL_VENDOR

Returns the company responsible for this EGL implementation. This name does not change from release to release.

EGL_VERSION

Returns a version or release number. The EGL_VERSION string is laid out as follows:

```
major_version.minor_version space vendor_specific_info
```

EGL_EXTENSIONS

Returns a space separated list of supported extensions to EGL.

에러 코드

NULL is returned on failure.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_PARAMETER is generated if name is not an accepted value.

4.2.6 OEMC_eglGetConfigs

설명

Query EGL configuration list

프로토타입

```
M_EGLBoolean OEMC_eglGetConfigs ( M_EGLDisplay dpy,  
M_EGLConfig* configs, M_EGLint config_size, M_EGLint* num_config )
```

매개변수

dpy

Specifies the EGL display connection.

configs

Returns a list of configs.

config_size

Specifies the size of the list of configs.

num_config

Returns the number of configs returned.

참고사항

eglGetConfigs returns a list of all EGL frame buffer configurations that are available for the specified display. The items in the list can be used in any EGL function that requires an EGL frame buffer configuration.

configs does not return values, if it is specified as NULL. This is useful for querying just the number of all frame buffer configurations.

Use eglGetConfigAttrib to retrieve individual attribute values of a frame buffer configuration.

예리 코드

EGL_FALSE is returned on failure, EGL_TRUE otherwise. configs and num_config are not modified when EGL_FALSE is returned.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_PARAMETER is generated if num_config is NULL.

4.2.7 OEMC_eglChooseConfig

설명

Selective query of EGL configuration list

프로토타입

```
M_EGLBoolean OEMC_eglChooseConfig ( M_EGLDisplay dpy, const  
M_EGLint* attrib_list, M_EGLConfig* configs, M_EGLint config_size, M_EGLint*  
num_config )
```

매개변수

display

Specifies the EGL display connection.

attrib_list

Specifies attributes required to match by configs.

configs

Returns an array of frame buffer configurations.

config_size

Specifies the size of the array of frame buffer configurations.

num_config

Returns the number of frame buffer configurations returned.

참고사항

eglChooseConfig returns a list of all EGL frame buffer configurations that match the attributes specified in attrib_list. The items in the list can be used in any EGL function that requires an EGL frame buffer configuration.

configs does not return values, if it is specified as NULL. This is useful for querying just the number of matching frame buffer configurations.

All attributes in attrib_list, including boolean attributes, are immediately followed by the corresponding desired value. The list is terminated with EGL_NONE. If an attribute is not

specified in attrib_list then the default value (see below) is used (and the attribute is said to be specified implicitly). For example, if EGL_DEPTH_SIZE is not specified then it is assumed to be 0. For some attributes, the default is EGL_DONT_CARE meaning that any value is OK for this attribute, so the attribute will not be checked.

Attributes are matched in an attribute-specific manner. Some of the attributes, such as EGL_LEVEL, must match the specified value exactly. Others, such as, EGL_RED_SIZE must meet or exceed the specified minimum values. If more than one EGL frame buffer configuration is found, then a list of configurations, sorted according to the "best" match criteria, is returned. The match criteria for each attribute and the exact sorting order is defined below.

The interpretations of the various EGL frame buffer configuration attributes are as follows:

EGL_BUFFER_SIZE

Must be followed by a nonnegative integer that indicates the desired color buffer size. The smallest color buffer of at least the specified size is preferred. The default value is 0.

EGL_RED_SIZE

Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available red buffer is preferred. Otherwise, the largest available red buffer of at least the minimum size is preferred. The default value is 0.

EGL_GREEN_SIZE

Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available green buffer is preferred. Otherwise, the largest available green buffer of at least the minimum size is preferred. The default value is 0.

EGL_BLUE_SIZE

Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available blue buffer is preferred. Otherwise, the largest available blue buffer of at least the minimum size is preferred. The default value is 0.

EGL_ALPHA_SIZE

Must be followed by a nonnegative minimum size specification. If this value is zero, the

smallest available alpha buffer is preferred. Otherwise, the largest available alpha buffer of at least the minimum size is preferred. The default value is 0.

EGL_CONFIG_CAVEAT

Must be followed by one of EGL_DONT_CARE, EGL_NONE, EGL_SLOW_CONFIG, EGL_NON_CONFORMANT_CONFIG. If EGL_NONE is specified, then only frame buffer configurations with no caveats will be considered. If EGL_SLOW_CONFIG is specified, then only slow frame buffer configurations will be considered. If EGL_NON_CONFORMANT_CONFIG is specified, then only non-conformant frame buffer configurations will be considered. The default value is EGL_DONT_CARE.

EGL_CONFIG_ID

Must be followed by a valid ID that indicates the desired EGL frame buffer configuration. When a EGL_CONFIG_ID is specified, all attributes are ignored. The default value is EGL_DONT_CARE.

EGL_DEPTH_SIZE

Must be followed by a nonnegative integer that indicates the desired depth buffer size. The smallest available depth buffer of at least the minimum size is preferred. If the desired value is zero, frame buffer configurations with no depth buffer are preferred. The default value is 0.

EGL_LEVEL

Must be followed by an integer buffer-level specification. This specification is honored exactly. Buffer level 0 corresponds to the default frame buffer of the display. Buffer level 1 is the first overlay frame buffer, level two the second overlay frame buffer, and so on. Negative buffer levels correspond to underlay frame buffers. The default value is 0.

EGL_NATIVE_RENDERABLE

Must be followed by EGL_DONT_CARE, EGL_TRUE, or EGL_FALSE. If EGL_TRUE is specified, then only frame buffer configurations that allow native rendering into the surface will be considered. The default value is EGL_DONT_CARE.

EGL_NATIVE_VISUAL_TYPE

Must be followed by a platform dependent value or EGL_DONT_CARE. The default value is EGL_DONT_CARE.

EGL_SAMPLE_BUFFERS

Must be followed by the minimum acceptable number of multisample buffers. Configurations with the smallest number of multisample buffers that meet or exceed this minimum number are preferred. Currently operation with more than one multisample buffer is undefined, so only values of zero or one will produce a match. The default value is 0.

EGL_SAMPLES

Must be followed by the minimum number of samples required in multisample buffers. Configurations with the smallest number of samples that meet or exceed the specified minimum number are preferred. Note that it is possible for color samples in the multisample buffer to have fewer bits than colors in the main color buffers. However, multisampled colors maintain at least as much color resolution in aggregate as the main color buffers.

EGL_STENCIL_SIZE

Must be followed by a nonnegative integer that indicates the desired number of stencil bitplanes. The smallest stencil buffer of at least the specified size is preferred. If the desired value is zero, frame buffer configurations with no stencil buffer are preferred. The default value is 0.

EGL_SURFACE_TYPE

Must be followed by a mask indicating which EGL surface types the frame buffer configuration must support. Valid bits are EGL_WINDOW_BIT, EGL_PBUFFER_BIT, and EGL_PIXMAP_BIT. For example, if mask is set to EGL_WINDOW_BIT | EGL_PIXMAP_BIT, only frame buffer configurations that support both windows and pixmaps will be considered. The default value is EGL_WINDOW_BIT.

EGL_TRANSPARENT_TYPE

Must be followed by one of EGL_NONE or EGL_TRANSPARENT_RGB. If EGL_NONE is specified, then only opaque frame buffer configurations will be considered. If EGL_TRANSPARENT_RGB is specified, then only transparent frame buffer configurations will be considered. The default value is EGL_NONE.

EGL_TRANSPARENT_RED_VALUE

Must be followed by an integer value indicating the transparent red value. The value must be between 0 and the maximum color buffer value for red. Only frame buffer configurations that use the specified transparent red value will be considered. The default value is EGL_DONT_CARE.

This attribute is ignored unless EGL_TRANSPARENT_TYPE is included in attrib_list and specified as EGL_TRANSPARENT_RGB.

EGL_TRANSPARENT_GREEN_VALUE

Must be followed by an integer value indicating the transparent green value. The value must be between 0 and the maximum color buffer value for red. Only frame buffer configurations that use the specified transparent green value will be considered. The default value is EGL_DONT_CARE.

This attribute is ignored unless EGL_TRANSPARENT_TYPE is included in attrib_list and specified as EGL_TRANSPARENT_RGB.

EGL_TRANSPARENT_BLUE_VALUE

Must be followed by an integer value indicating the transparent blue value. The value must be between 0 and the maximum color buffer value for red. Only frame buffer configurations that use the specified transparent blue value will be considered. The default value is EGL_DONT_CARE.

This attribute is ignored unless EGL_TRANSPARENT_TYPE is included in attrib_list and specified as EGL_TRANSPARENT_RGB.

에러 코드

EGL_FALSE is returned on failure, EGL_TRUE otherwise. configs and num_config are not modified when EGL_FALSE is returned.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_BAD_ATTRIBUTE is generated if attribute_list contains an invalid frame buffer configuration attribute or an attribute value that is unrecognized or out of range.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_PARAMETER is generated if num_config is NULL.

4.2.8 OEMC_eglGetConfigAttrib

설명

Query EGL configuration attribute value

프로토타입

M_EGLBoolean OEMC_eglGetConfigAttrib (M_EGLDisplay dpy, M_EGLConfig config, M_EGLint attribute, M_EGLint* value)

매개변수

display

Specifies the EGL display connection.

config

Specifies the EGL frame buffer configuration to be queried.

attribute

Specifies the EGL rendering context attribute to be returned.

value

Returns the requested value.

참고사항

eglGetConfigAttrib returns in value the value of attribute for config. attribute can be one of the following:

EGL_BUFFER_SIZE

Returns the depth of the color buffer. It is the sum of EGL_RED_SIZE, EGL_GREEN_SIZE, EGL_BLUE_SIZE, and EGL_ALPHA_SIZE.

EGL_RED_SIZE

Returns the number of bits of red stored in the color buffer.

EGL_GREEN_SIZE

Returns the number of bits of green stored in the color buffer.

EGL_BLUE_SIZE

Returns the number of bits of blue stored in the color buffer.

EGL_ALPHA_SIZE

Returns the number of bits of alpha stored in the color buffer.

EGL_CONFIG_CAVEAT

Returns the caveats for the frame buffer configuration. Possible caveat values are EGL_NONE, EGL_SLOW_CONFIG, and EGL_NON_CONFORMANT.

EGL_CONFIG_ID

Returns the ID of the frame buffer configuration.

EGL_DEPTH_SIZE

Returns the number of bits in the depth buffer.

EGL_LEVEL

Returns the frame buffer level. Level zero is the default frame buffer. Positive levels correspond to frame buffers that overlay the default buffer and negative levels correspond to frame buffers that underlay the default buffer.

EGL_MAX_PBUFFER_WIDTH

Returns the maximum width of a pixel buffer surface in pixels.

EGL_MAX_PBUFFER_HEIGHT

Returns the maximum height of a pixel buffer surface in pixels.

EGL_MAX_PBUFFER_PIXELS

Returns the maximum size of a pixel buffer surface in pixels.

EGL_NATIVE_RENDERABLE

Returns EGL_TRUE if native rendering APIs can render into the surface, EGL_FALSE otherwise.

EGL_NATIVE_VISUAL_ID

Returns the ID of the associated native visual.

EGL_NATIVE_VISUAL_TYPE

Returns the type of the associated native visual.

EGL_PRESERVED_RESOURCES

Returns EGL_TRUE if resources are preserved across power management events, EGL_FALSE otherwise.

EGL_SAMPLE_BUFFERS

Returns the number of multisample buffers.

EGL_SAMPLES

Returns the number of samples per pixel.

EGL_STENCIL_BITS

Returns the number of bits in the stencil buffer.

EGL_SURFACE_TYPE

Returns the types of supported EGL surfaces.

EGL_TRANSPARENT_TYPE

Returns the type of supported transparency. Possible transparency values are: EGL_NONE, and EGL_TRANSPARENT_RGB.

EGL_TRANSPARENT_RED

Returns the transparent red value.

EGL_TRANSPARENT_GREEN

Returns the transparent green value.

EGL_TRANSPARENT_BLUE

Returns the transparent blue value.

예러 코드

EGL_FALSE is returned on failure, EGL_TRUE otherwise. value is not modified when

EGL_FALSE is returned.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONFIG is generated if config is not an EGL frame buffer configuration.

EGL_BAD_ATTRIBUTE is generated if attribute is not a valid frame buffer configuration attribute.

4.2.9 OEMC_eglCreateWindowSurface

설명

Create Native Window Surface

프로토타입

```
M_EGLSurface OEMC_eglCreateWindowSurface ( M_EGLDisplay dpy,  
M_EGLConfig config, M_NativeWindowType win, const M_EGLint* attrib_list )
```

매개변수

display

Specifies the EGL display connection.

config

Specifies the EGL frame buffer configuration that defines the frame buffer resource available to the surface.

native_window

Specifies the native window.

attrib_list

Specifies window surface attributes. Must be NULL or empty (first attribute is EGL_NONE).

참고사항

eglCreateWindowSurface creates an EGL window surface and returns its handle. If eglCreateWindowSurface fails to create a window surface, EGL_NO_SURFACE is returned.

Any EGL rendering context that was created with respect to config can be used to render into the surface. Use eglMakeCurrent to attach an EGL rendering context to the surface.

Use eglQuerySurface to retrieve the ID of config.

Use eglDestroySurface to destroy the surface.

This function always return EGL_NO_SURFACE in GIGA 3D.

에러 코드

EGL_NO_SURFACE is returned if creation of the context fails.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONFIG is generated if config is not an EGL frame buffer configuration.

EGL_BAD_NATIVE_WINDOW may be generated if native_window is not a valid native window.

EGL_BAD_ATTRIBUTE is generated if attrib_list contains an invalid window attribute or if an attribute value is not recognized or is out of range.

EGL_BAD_ALLOC is generated if there are not enough resources to allocate the new surface.

EGL_BAD_MATCH is generated if the attributes of native_window do not correspond to config or if config does not support rendering to windows (the EGL_SURFACE_TYPE attribute does not contain EGL_WINDOW_BIT).

4.2.10 OEMC_eglCreatePixmapSurface

설명

Create Native Pixmap Surface

프로토타입

```
M_EGLSurface OEMC_eglCreatePixmapSurface ( M_EGLDisplay dpy, M_EGLConfig config, M_NativePixmapType pixmap, const M_EGLint* attrib_list )
```

매개변수

display

Specifies the EGL display connection.

config

Specifies the EGL frame buffer configuration that defines the frame buffer resource available to the surface.

native_pixmap

Specifies the native pixmap. HIMAGE of GIGA 2D

attrib_list

Specifies pixmap surface attributes. Must be NULL or empty (first attribute is EGL_NONE).

참고사항

eglCreatePixmapSurface creates an off-screen EGL pixmap surface and returns its handle. If eglCreatePixmapSurface fails to create a pixmap surface, EGL_NO_SURFACE is returned.

Any EGL rendering context that was created with respect to config can be used to render into the surface. Use eglMakeCurrent to attach an EGL rendering context to the surface.

Use eglQuerySurface to retrieve the ID of config.

Use eglDestroySurface to destroy the surface.

에러 코드

EGL_NO_SURFACE is returned if creation of the context fails.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONFIG is generated if config is not an EGL config.

EGL_BAD_NATIVE_PIXMAP may be generated if native_pixmap is not a valid native pixmap.

EGL_BAD_ATTRIBUTE is generated if attrib_list contains an invalid pixmap attribute or if an attribute value is not recognized or out of range.

EGL_BAD_ALLOC is generated if there are not enough resources to allocate the new surface.

EGL_BAD_MATCH is generated if the attributes of native_pixmap do not correspond to config or if config does not support rendering to pixmaps (the EGL_SURFACE_TYPE attribute does not contain EGL_PIXMAP_BIT).

4.2.11 OEMC_eglCreatePbufferSurface

설명

Create pBuff surface of OES system

프로토타입

```
M_EGLSurface OEMC_eglCreatePbufferSurface ( M_EGLDisplay dpy, M_EGLConfig  
config, const M_EGLint* attrib_list )
```

매개변수

display

Specifies the EGL display connection.

config

Specifies the EGL frame buffer configuration that defines the frame buffer resource available to the surface.

attrib_list

Specifies the pixel buffer surface attributes. May be NULL or empty (first attribute is EGL_NONE). Accepted attributes are EGL_WIDTH, EGL_HEIGHT, and EGL_LARGEST_PBUFFER.

참고사항

eglCreatePbufferSurface creates an off-screen pixel buffer surface and returns its handle. If eglCreatePbufferSurface fails to create a pixel buffer surface, EGL_NO_SURFACE is returned.

Any EGL rendering context that was created with respect to config can be used to render into the surface. Use eglMakeCurrent to attach an EGL rendering context to the surface.

Use eglQuerySurface to retrieve the dimensions of the allocated pixel buffer surface or the ID of config.

Use eglDestroySurface to destroy the surface.

The pixel buffer surface attributes are specified as a list of attribute value pairs,

terminated with EGL_NONE. The accepted attributes for an EGL pixel buffer surface are:

EGL_WIDTH

Requests a pixel buffer surface with the specified width. The default value is 0.

EGL_HEIGHT

Requests a pixel buffer surface with the specified height. The default value is 0.

EGL_LARGEST_PBUFFER

Requests a pixel buffer surface with the largest width and height. Use `eglQuerySurface` to retrieve the dimensions of the allocated pixel buffer. Default value is EGL_FALSE.

에러 코드

EGL_NO_SURFACE is returned if creation of the context fails.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONFIG is generated if config is not an EGL frame buffer configuration.

EGL_BAD_ATTRIBUTE is generated if attrib_list contains an invalid pixel buffer attribute or if an attribute value is not recognized or out of range.

EGL_BAD_ALLOC is generated if there are not enough resources to allocate the new surface.

EGL_BAD_MATCH is generated if config does not support rendering to pixel buffers (the EGL_SURFACE_TYPE attribute does not contain EGL_PBUFFER_BIT).

4.2.12 OEMC_eglDestroySurface

설명

Destory surface

프로토타입

```
M_EGLBoolean OEMC_eglDestroySurface ( M_EGLDisplay dpy,  
M_EGLSurface surface )
```

매개변수

display

Specifies the EGL display connection.

surface

Specifies the EGL surface to be destroyed.

참고사항

If the EGL surface surface is not current to any thread, eglDestroySurface destroys it immediately. Otherwise, surface is destroyed when it becomes not current to any thread.

에러 코드

EGL_FALSE is returned if destruction of the surface fails, EGL_TRUE otherwise.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_SURFACE is generated if surface is not an EGL surface.

4.2.13 OEMC_eglQuerySurface

설명

Query attribute of surface

프로토타입

M_EGLBoolean OEMC_eglQuerySurface (M_EGLDisplay dpy, M_EGLSurface surface,
M_EGLint attribute, M_EGLint* value)

매개변수

display

Specifies the EGL display connection.

surface

Specifies the EGL surface to query.

attribute

Specifies the EGL surface attribute to be returned.

value

Returns the requested value.

참고사항

eglQuerySurface returns in value the value of attribute for surface. attribute can be one of the following:

EGL_CONFIG_ID

Returns the ID of the EGL frame buffer configuration with respect to which the surface was created.

EGL_WIDTH

Returns the width of the surface in pixels.

EGL_HEIGHT

Returns the height of the surface in pixels.

EGL_LARGEST_PBUFFER

Returns the same attribute value specified when the surface was created with `eglCreatePbufferSurface`. For a window or pixmap surface, value is not modified.

에러 코드

`EGL_FALSE` is returned on failure, `EGL_TRUE` otherwise. value is not modified when `EGL_FALSE` is returned.

`EGL_BAD_DISPLAY` is generated if display is not an EGL display connection.

`EGL_NOT_INITIALIZED` is generated if display has not been initialized.

`EGL_BAD_SURFACE` is generated if surface is not an EGL surface.

`EGL_BAD_ATTRIBUTE` is generated if attribute is not a valid surface attribute.

4.2.14 OEMC_eglCreateContext

설명

Create context

프로토타입

```
M_EGLContext OEMC_eglCreateContext ( M_EGLDisplay dpy, M_EGLConfig  
config, M_EGLContext share_list, const M_EGLint* attrib_list )
```

매개변수

dpy

Specifies the EGL display connection.

config

Specifies the EGL frame buffer configuration that defines the frame buffer resource available to the rendering context.

share_context

Specifies the EGL rendering context with which to share texture objects. EGL_NO_CONTEXT indicates that no sharing is to take place.

attrib_list

Specifies attributes.

참고사항

eglCreateContext creates an EGL rendering context and returns its handle. This context can be used to render into an EGL drawing surface. If eglCreateContext fails to create a rendering context, EGL_NO_CONTEXT is returned.

If share_context is not EGL_NO_CONTEXT, then all texture objects except object 0, are shared by context share_context and by the newly created context. An arbitrary number of rendering contexts can share a single texture object space. However, all rendering contexts that share a single texture object space must themselves exist in the same address space. Two rendering contexts share an address space if both are owned by a single process.

Notes

A process is a single execution environment, implemented in a single address space, consisting of one or more threads.

A thread is one of a set of subprocesses that share a single address space, but maintain separate program counters, stack spaces, and other related global data. A thread is the only member of its subprocess group is equivalent to a process.

에러 코드

EGL_NO_CONTEXT is returned if creation of the context fails.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONFIG is generated if config is not an EGL frame buffer configuration.

EGL_BAD_CONTEXT is generated if share_context is not an EGL rendering context and is not EGL_NO_CONTEXT.

EGL_BAD_ATTRIBUTE is generated if attrib_list contains an invalid context attribute or if an attribute is not recognized or out of range.

EGL_BAD_ALLOC is generated if there are not enough resources to allocate the new context.

4.2.15 OEMC_eglDestroyContext

설명

Destroy context

프로토타입

```
M_EGLBoolean OEMC_eglDestroyContext ( M_EGLDisplay dpy,  
M_EGLContext ctx )
```

매개변수

dpy

Specifies the EGL display connection.

context

Specifies the EGL rendering context to be destroyed.

참고사항

If the EGL rendering context context is not current to any thread, eglDestroyContext destroys it immediately. Otherwise, context is destroyed when it becomes not current to any thread.

에러 코드

EGL_FALSE is returned if destruction of the context fails, EGL_TRUE otherwise.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONTEXT is generated if context is not an EGL rendering context.

4.2.16 OEMC_eglMakeCurrent

설명

Make current context

프로토타입

M_EGLBoolean OEMC_eglMakeCurrent (M_EGLDisplay dpy, M_EGLSurface adraw, M_EGLSurface aread, M_EGLContext ctx)

매개변수

dpy

Specifies the EGL display connection.

draw

Specifies the EGL draw surface.

read

Specifies the EGL read surface.

context

Specifies the EGL rendering context to be attached to the surfaces.

참고사항

eglMakeCurrent binds context to the current rendering thread and to the draw and read surfaces. draw is used for all GL operations except for any pixel data read back (glReadPixels, glCopyTexImage2D, and glCopyTexSubImage2D), which is taken from the frame buffer values of read.

If the calling thread has already a current rendering context, that context is flushed and marked as no longer current.

The first time that context is made current, the viewport and scissor dimensions are set to the size of the draw surface. The viewport and scissor are not modified when context is subsequently made current.

To release the current context without assigning a new one, call eglMakeCurrent with

draw and read set to EGL_NO_SURFACE and context set to EGL_NO_CONTEXT.

Use `eglGetCurrentContext`, `eglGetCurrentDisplay`, and `eglGetCurrentSurface` to query the current rendering context and associated display connection and surfaces.

에러 코드

EGL_FALSE is returned on failure, EGL_TRUE otherwise. If EGL_FALSE is returned, the previously current rendering context and surfaces (if any) remain unchanged.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_SURFACE is generated if draw or read is not an EGL surface.

EGL_BAD_CONTEXT is generated if context is not an EGL rendering context.

EGL_BAD_MATCH is generated if draw or read are not compatible with context, or if context is set to EGL_NO_CONTEXT and draw or read are not set to EGL_NO_SURFACE, or if draw or read are set to EGL_NO_SURFACE and context is not set to EGL_NO_CONTEXT.

EGL_BAD_ACCESS is generated if context is current to some other thread.

EGL_BAD_NATIVE_PIXMAP may be generated if a native pixmap underlying either draw or read is no longer valid.

EGL_BAD_NATIVE_WINDOW may be generated if a native window underlying either draw or read is no longer valid.

EGL_BAD_CURRENT_SURFACE is generated if the previous context has unflushed commands and the previous surface is no longer valid.

EGL_BAD_ALLOC may be generated if allocation of ancillary buffers for draw or read were delayed until `eglMakeCurrent` is called, and there are not enough resources to allocate them.

4.2.17 OEMC_eglGetCurrentContext

설명

Query currently selected context

프로토타입

M_EGLContext OEMC_eglGetCurrentContext (void)

참고사항

eglGetCurrentContext returns the current EGL rendering context, as specified by eglMakeCurrent. If no context is current, EGL_NO_CONTEXT is returned.

4.2.18 OEMC_eglGetCurrentSurface

설명

Query currently linked surface

프로토타입

M_EGLSurface OEMC_eglGetCurrentSurface (M_EGLint readdraw)

매개변수

readdraw

Specifies whether the EGL read or draw surface is to be returned.

참고사항

eglGetCurrentSurface returns the read or draw surface attached to the current EGL rendering context, as specified by eglMakeCurrent. If no context is current, EGL_NO_SURFACE is returned.

4.2.19 OEMC_eglGetCurrentDisplay

설명

Query currently linked display

프로토타입

M_EGLDisplay OEMC_eglGetCurrentDisplay (void)

참고사항

eglGetCurrentDisplay returns the current EGL display connection for the current EGL rendering context, as specified by eglMakeCurrent. If no context is current, EGL_NO_DISPLAY is returned.

4.2.20 OEMC_eglQueryContext

설명

Query attribute of context

프로토타입

```
M_EGLBoolean OEMC_eglQueryContext ( M_EGLDisplay dpy,  
M_EGLContext ctx, M_EGLint attribute, M_EGLint* value )
```

매개변수

dpy

Specifies the EGL display connection.

ctx

Specifies the EGL rendering context to query.

attribute

Specifies the EGL rendering context attribute to be returned.

value

Returns the requested value.

참고사항

eglQueryContext returns in value the value of attribute for context. attribute can be one of the following:

EGL_CONFIG_ID

Returns the ID of the EGL frame buffer configuration with respect to which the context was created.

에러 코드

EGL_FALSE is returned on failure, EGL_TRUE otherwise. value is not modified when EGL_FALSE is returned.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_CONTEXT is generated if context is not an EGL rendering context.

EGL_BAD_ATTRIBUTE is generated if attribute is not a valid context attribute.

4.2.21 OEMC_eglWaitGL

설명

Wait until OES operation is done

프로토타입

M_EGLBoolean OEMC_eglWaitGL (void)

참고사항

GL rendering calls made prior to eglWaitGL are guaranteed to be executed before native rendering calls made after eglWaitGL. The same result can be achieved using glFinish.

eglWaitGL is ignored if there is no current EGL rendering context.

에러 코드

EGL_FALSE is returned if eglWaitGL fails, EGL_TRUE otherwise.

EGL_BAD_NATIVE_SURFACE is generated if the surface associated with the current context has a native window or pixmap, and that window or pixmap is no longer valid.

4.2.22 OEMC_eglWaitNative

설명

Wait until Native system operation is done

프로토타입

M_EGLBoolean OEMC_eglWaitNative (M_EGLint engine)

매개변수

engine

Specifies a particular marking engine to be waited on. Must be EGL_EGL_CORE_NATIVE_ENGINE.

참고사항

Native rendering calls made prior to eglWaitNative are guaranteed to be executed before GL rendering calls made after eglWaitNative.

eglWaitNative is ignored if there is no current EGL rendering context.

에러 코드

EGL_BAD_PARAMETER is generated if engine is not a recognized marking engine.

EGL_BAD_NATIVE_SURFACE is generated if the surface associated with the current context has a native window or pixmap, and that window or pixmap is no longer valid.

4.2.23 OEMC_eglSwapBuffers

설명

Swap buffer contents

프로토타입

```
M_EGLBoolean OEMC_eglSwapBuffers ( M_EGLDisplay dpy,  
M_EGLSurface surface )
```

매개변수

dpy

Specifies the EGL display connection.

surface

Specifies the EGL drawing surface whose buffers are to be swapped.

참고사항

If surface is a window surface, eglSwapBuffers posts its color buffer to the associated native window.

eglSwapBuffers performs an implicit glFlush before it returns. Subsequent GL commands may be issued immediately after calling eglSwapBuffers, but are not executed until the buffer exchange is completed.

If surface is a pixel buffer or a pixmap, eglSwapBuffers has no effect, and no error is generated.

에러 코드

EGL_FALSE is returned if swapping of the surface buffers fails, EGL_TRUE otherwise.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_SURFACE is generated if surface is not an EGL drawing surface.

4.2.24 OEMC_eglCopyBuffers

설명

Copy surface contents to target pixmap

프로토타입

```
M_EGLBoolean OEMC_eglCopyBuffers ( M_EGLDisplay dpy,  
M_EGLSurface surface, M_NativePixmapType target )
```

매개변수

dpy

Specifies the EGL display connection.

surface

Specifies the EGL surface whose color buffer is to be copied.

native_pixmap

Specifies the native pixmap as target of the copy.

참고사항

eglCopyBuffers copies the color buffer of surface to native_pixmap.

eglCopyBuffers performs an implicit glFlush before it returns. Subsequent GL commands may be issued immediately after calling eglCopyBuffers, but are not executed until copying of the color buffer is completed.

에러 코드

EGL_FALSE is returned if swapping of the surface buffers fails, EGL_TRUE otherwise.

EGL_BAD_DISPLAY is generated if display is not an EGL display connection.

EGL_NOT_INITIALIZED is generated if display has not been initialized.

EGL_BAD_SURFACE is generated if surface is not an EGL drawing surface.

EGL_BAD_NATIVE_PIXMAP is generated if the implementation does not support native pixmaps.

EGL_BAD_NATIVE_PIXMAP may be generated if native_pixmap is not a valid native pixmap.

EGL_BAD_MATCH is generated if the format of native_pixmap is not compatible with the color buffer of surface.

4.2.25 OEMC_eglGetProcAddress

설명

return a GL or an EGL extension function

프로토타입

```
void ( *OEMC_eglGetProcAddress( const char *procname ) ) ( void )
```

매개변수

procname

[in]

Specifies the name of the function to return.

참고 사항

eglGetProcAddress returns the address of the extension function named by procname. procname must be a null-terminated string. The pointer returned should be cast to a function pointer type matching the extension function's definition in that extension specification. A return value of NULL indicates that the specific function does not exist for the EGL implementation.

A non-NULL return value does not guarantee that an extension function is actually supported at runtime. The client must also query glGetString(GL_EXTENSIONS) or eglQueryString(display, EGL_EXTENSIONS) to determine if an extension is supported by a particular context or display.

Function pointers returned by eglGetProcAddress are independent of the display and the currently bound context and may be used by any context which supports the extension.

eglGetProcAddress may be queried for all GL and EGL extension functions.

See Also

glGetString, eglQueryString