NAME

Direct3D 10 Interoperability [DEPRECATED] −

Enumerations

enum **cudaD3D10MapFlags** { **cudaD3D10MapFlagsNone** = 0, **cudaD3D10MapFlagsReadOnly** = 1, **cudaD3D10MapFlagsWriteDiscard** = 2 }

enum **cudaD3D10RegisterFlags** { **cudaD3D10RegisterFlagsNone** = 0, **cudaD3D10RegisterFlagsArray** = 1 }

Functions

cudaError_t cudaD3D10GetDirect3DDevice (ID3D10Device **ppD3D10Device)
Gets the Direct3D device against which the current CUDA context was created.

cudaError_t cudaD3D10MapResources (int count, ID3D10Resource **ppResources)
Maps Direct3D Resources for access by CUDA.

cudaError_t cudaD3D10RegisterResource (ID3D10Resource *pResource, unsigned int flags)
Registers a Direct3D 10 resource for access by CUDA.

cudaError_t cudaD3D10ResourceGetMappedArray (cudaArray **ppArray, ID3D10Resource *pResource, unsigned int subResource)
Gets an array through which to access a subresource of a Direct3D resource which has been mapped for access by CUDA.

cudaError_t cudaD3D10ResourceGetMappedPitch (size_t *pPitch, size_t *pPitchSlice, ID3D10Resource *pResource, unsigned int subResource)
Gets the pitch of a subresource of a Direct3D resource which has been mapped for access by CUDA.

cudaError_t cudaD3D10ResourceGetMappedPointer (void **pPointer, ID3D10Resource *pResource, unsigned int subResource)
Gets a pointer through which to access a subresource of a Direct3D resource which has been mapped for access by CUDA.

cudaError_t cudaD3D10ResourceGetMappedSize (size_t *pSize, ID3D10Resource *pResource, unsigned int subResource)
Gets the size of a subresource of a Direct3D resource which has been mapped for access by CUDA.

cudaError_t cudaD3D10ResourceGetSurfaceDimensions (size_t *pWidth, size_t *pHeight, size_t *pDepth, ID3D10Resource *pResource, unsigned int subResource)
Gets the dimensions of a registered Direct3D surface.

cudaError_t cudaD3D10ResourceSetMapFlags (ID3D10Resource *pResource, unsigned int flags)
Set usage flags for mapping a Direct3D resource.

cudaError_t cudaD3D10SetDirect3DDevice (ID3D10Device *pD3D10Device, int device=-1)
Sets the Direct3D 10 device to use for interoperability with a CUDA device.

cudaError_t cudaD3D10UnmapResources (int count, ID3D10Resource **ppResources)
Unmaps Direct3D resources.

cudaError_t cudaD3D10UnregisterResource (ID3D10Resource *pResource)
Unregisters a Direct3D resource.

Detailed Description

This section describes deprecated Direct3D 10 interoperability functions.

Enumeration Type Documentation

enum cudaD3D10MapFlags

CUDA D3D10 Map Flags

Enumerator:

cudaD3D10MapFlagsNone
Default; Assume resource can be read/written

cudaD3D10MapFlagsReadOnly
CUDA kernels will not write to this resource

cudaD3D10MapFlagsWriteDiscard
CUDA kernels will only write to and will not read from this resource

enum cudaD3D10RegisterFlags

CUDA D3D10 Register Flags

Enumerator:

cudaD3D10RegisterFlagsNone
: Default; Resource can be accessed through a void*

cudaD3D10RegisterFlagsArray
: Resource can be accessed through a CUarray*

Function Documentation

cudaError_t cudaD3D10GetDirect3DDevice (ID3D10Device ** ppD3D10Device)

Deprecated

This function is deprecated as of CUDA 5.0.

This function is deprecated and should no longer be used. It is no longer necessary to associate a CUDA device with a D3D10 device in order to achieve maximum interoperability performance.

Parameters:

ppD3D10Device - Returns the Direct3D device for this thread

Returns:

cudaSuccess, **cudaErrorUnknown**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaD3D10SetDirect3DDevice

cudaError_t cudaD3D10MapResources (int count, ID3D10Resource ** ppResources)

Deprecated

This function is deprecated as of CUDA 3.0.

Maps the `count` Direct3D resources in `ppResources` for access by CUDA.

The resources in `ppResources` may be accessed in CUDA kernels until they are unmapped. Direct3D should not access any resources while they are mapped by CUDA. If an application does so, the results are undefined.

This function provides the synchronization guarantee that any Direct3D calls issued before **cudaD3D10MapResources()** will complete before any CUDA kernels issued after **cudaD3D10MapResources()** begin.

If any of `ppResources` have not been registered for use with CUDA or if `ppResources` contains any duplicate entries then **cudaErrorInvalidResourceHandle** is returned. If any of `ppResources` are presently mapped for access by CUDA then **cudaErrorUnknown** is returned.

Parameters:

count - Number of resources to map for CUDA
ppResources - Resources to map for CUDA

Returns:

cudaSuccess, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsMapResources

cudaError_t cudaD3D10RegisterResource (ID3D10Resource * pResource, unsigned int flags)

Deprecated

This function is deprecated as of CUDA 3.0.

Registers the Direct3D resource `pResource` for access by CUDA.

If this call is successful, then the application will be able to map and unmap this resource until it is unregistered through **cudaD3D10UnregisterResource()**. Also on success, this call will increase the internal reference count on `pResource`. This reference count will be decremented when this resource

is unregistered through **cudaD3D10UnregisterResource()**.

This call potentially has a high-overhead and should not be called every frame in interactive applications.

The type of `pResource` must be one of the following:

- ID3D10Buffer: Cannot be used with `flags` set to `cudaD3D10RegisterFlagsArray`.
- ID3D10Texture1D: No restrictions.
- ID3D10Texture2D: No restrictions.
- ID3D10Texture3D: No restrictions.

The `flags` argument specifies the mechanism through which CUDA will access the Direct3D resource. The following values are allowed.

- **cudaD3D10RegisterFlagsNone**: Specifies that CUDA will access this resource through a `void*`. The pointer, size, and pitch for each subresource of this resource may be queried through **cudaD3D10ResourceGetMappedPointer()**, **cudaD3D10ResourceGetMappedSize()**, and **cudaD3D10ResourceGetMappedPitch()** respectively. This option is valid for all resource types.
- **cudaD3D10RegisterFlagsArray**: Specifies that CUDA will access this resource through a `CUarray` queried on a sub-resource basis through **cudaD3D10ResourceGetMappedArray()**. This option is only valid for resources of type ID3D10Texture1D, ID3D10Texture2D, and ID3D10Texture3D.

Not all Direct3D resources of the above types may be used for interoperability with CUDA. The following are some limitations.

- The primary rendertarget may not be registered with CUDA.
- Resources allocated as shared may not be registered with CUDA.
- Textures which are not of a format which is 1, 2, or 4 channels of 8, 16, or 32-bit integer or floating-point data cannot be shared.
- Surfaces of depth or stencil formats cannot be shared.

If Direct3D interoperability is not initialized on this context then **cudaErrorInvalidDevice** is returned. If `pResource` is of incorrect type or is already registered then **cudaErrorInvalidResourceHandle** is returned. If `pResource` cannot be registered then **cudaErrorUnknown** is returned.

Parameters:
pResource - Resource to register
flags - Parameters for resource registration

Returns:
cudaSuccess, **cudaErrorInvalidDevice**, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:
Note that this function may also return error codes from previous, asynchronous launches.

See also:
cudaGraphicsD3D10RegisterResource

cudaError_t cudaD3D10ResourceGetMappedArray (cudaArray ** ppArray, ID3D10Resource * pResource, unsigned int subResource)
Deprecated
This function is deprecated as of CUDA 3.0.

Returns in `*ppArray` an array through which the subresource of the mapped Direct3D resource `pResource` which corresponds to `subResource` may be accessed. The value set in `ppArray` may change every time that `pResource` is mapped.

If `pResource` is not registered, then **cudaErrorInvalidResourceHandle** is returned. If `pResource` was not registered with usage flags **cudaD3D10RegisterFlagsArray**, then **cudaErrorInvalidResourceHandle** is returned. If `pResource` is not mapped then **cudaErrorUnknown** is returned.

For usage requirements of the `subResource` parameter, see **cudaD3D10ResourceGetMappedPointer()**.

Parameters:

ppArray - Returned array corresponding to subresource
pResource - Mapped resource to access
subResource - Subresource of pResource to access

Returns:

cudaSuccess, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsSubResourceGetMappedArray

cudaError_t cudaD3D10ResourceGetMappedPitch (size_t * pPitch, size_t * pPitchSlice, ID3D10Resource * pResource, unsigned int subResource)
Deprecated

This function is deprecated as of CUDA 3.0.

Returns in `*pPitch` and `*pPitchSlice` the pitch and Z-slice pitch of the subresource of the mapped Direct3D resource `pResource`, which corresponds to `subResource`. The values set in `pPitch` and `pPitchSlice` may change every time that `pResource` is mapped.

The pitch and Z-slice pitch values may be used to compute the location of a sample on a surface as follows.

For a 2D surface, the byte offset of the sample at position **x**, **y** from the base pointer of the surface is:

y * **pitch** + (**bytes per pixel**) * **x**

For a 3D surface, the byte offset of the sample at position **x**, **y**, **z** from the base pointer of the surface is:

z* **slicePitch** + **y** * **pitch** + (**bytes per pixel**) * **x**

Both parameters `pPitch` and `pPitchSlice` are optional and may be set to NULL.

If `pResource` is not of type ID3D10Texture1D, ID3D10Texture2D, or ID3D10Texture3D, or if `pResource` has not been registered for use with CUDA, then **cudaErrorInvalidResourceHandle** is returned. If `pResource` was not registered with usage flags **cudaD3D10RegisterFlagsNone**, then **cudaErrorInvalidResourceHandle** is returned. If `pResource` is not mapped for access by CUDA then **cudaErrorUnknown** is returned.

For usage requirements of the `subResource` parameter see **cudaD3D10ResourceGetMappedPointer()**.

Parameters:

pPitch - Returned pitch of subresource
pPitchSlice - Returned Z-slice pitch of subresource
pResource - Mapped resource to access
subResource - Subresource of pResource to access

Returns:

cudaSuccess, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsSubResourceGetMappedArray

cudaError_t cudaD3D10ResourceGetMappedPointer (void ** pPointer, ID3D10Resource * pResource, unsigned int subResource)
Deprecated

This function is deprecated as of CUDA 3.0.

Returns in `*pPointer` the base pointer of the subresource of the mapped Direct3D resource

pResource which corresponds to subResource. The value set in pPointer may change every time that pResource is mapped.

If pResource is not registered, then **cudaErrorInvalidResourceHandle** is returned. If pResource was not registered with usage flags **cudaD3D9RegisterFlagsNone**, then **cudaErrorInvalidResourceHandle** is returned. If pResource is not mapped then **cudaErrorUnknown** is returned.

If pResource is of type ID3D10Buffer then subResource must be 0. If pResource is of any other type, then the value of subResource must come from the subresource calculation in D3D10CalcSubResource().

Parameters:

pPointer - Returned pointer corresponding to subresource
pResource - Mapped resource to access
subResource - Subresource of pResource to access

Returns:

cudaSuccess, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsResourceGetMappedPointer

cudaError_t cudaD3D10ResourceGetMappedSize (size_t * pSize, ID3D10Resource * pResource, unsigned int subResource)
Deprecated

This function is deprecated as of CUDA 3.0.

Returns in *pSize the size of the subresource of the mapped Direct3D resource pResource which corresponds to subResource. The value set in pSize may change every time that pResource is mapped.

If pResource has not been registered for use with CUDA then cudaErrorInvalidHandle is returned. If pResource was not registered with usage flags **cudaD3D10RegisterFlagsNone**, then **cudaErrorInvalidResourceHandle** is returned. If pResource is not mapped for access by CUDA then **cudaErrorUnknown** is returned.

For usage requirements of the subResource parameter see **cudaD3D10ResourceGetMappedPointer()**.

Parameters:

pSize - Returned size of subresource
pResource - Mapped resource to access
subResource - Subresource of pResource to access

Returns:

cudaSuccess, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsResourceGetMappedPointer

cudaError_t cudaD3D10ResourceGetSurfaceDimensions (size_t * pWidth, size_t * pHeight, size_t * pDepth, ID3D10Resource * pResource, unsigned int subResource)
Deprecated

This function is deprecated as of CUDA 3.0.

Returns in *pWidth, *pHeight, and *pDepth the dimensions of the subresource of the mapped Direct3D resource pResource which corresponds to subResource.

Since anti-aliased surfaces may have multiple samples per pixel, it is possible that the dimensions of a resource will be an integer factor larger than the dimensions reported by the Direct3D runtime.

The parameters `pWidth`, `pHeight`, and `pDepth` are optional. For 2D surfaces, the value returned in `*pDepth` will be 0.

If `pResource` is not of type ID3D10Texture1D, ID3D10Texture2D, or ID3D10Texture3D, or if `pResource` has not been registered for use with CUDA, then cudaErrorInvalidHandle is returned.

For usage requirements of `subResource` parameters see **cudaD3D10ResourceGetMappedPointer()**.

Parameters:

pWidth - Returned width of surface
pHeight - Returned height of surface
pDepth - Returned depth of surface
pResource - Registered resource to access
subResource - Subresource of pResource to access

Returns:

cudaSuccess, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**,

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsSubResourceGetMappedArray

cudaError_t cudaD3D10ResourceSetMapFlags (ID3D10Resource * pResource, unsigned int flags)
Deprecated

This function is deprecated as of CUDA 3.0.

Set usage flags for mapping the Direct3D resource `pResource`.

Changes to flags will take effect the next time `pResource` is mapped. The `flags` argument may be any of the following:

- **cudaD3D10MapFlagsNone**: Specifies no hints about how this resource will be used. It is therefore assumed that this resource will be read from and written to by CUDA kernels. This is the default value.
- **cudaD3D10MapFlagsReadOnly**: Specifies that CUDA kernels which access this resource will not write to this resource.
- **cudaD3D10MapFlagsWriteDiscard**: Specifies that CUDA kernels which access this resource will not read from this resource and will write over the entire contents of the resource, so none of the data previously stored in the resource will be preserved.

If `pResource` has not been registered for use with CUDA then cudaErrorInvalidHandle is returned. If `pResource` is presently mapped for access by CUDA then **cudaErrorUnknown** is returned.

Parameters:

pResource - Registered resource to set flags for
flags - Parameters for resource mapping

Returns:

cudaSuccess, **cudaErrorInvalidValue**, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**,

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cudaGraphicsResourceSetMapFlags

cudaError_t cudaD3D10SetDirect3DDevice (ID3D10Device * pD3D10Device, int device = -1)
Deprecated

This function is deprecated as of CUDA 5.0.

This function is deprecated and should no longer be used. It is no longer necessary to associate a CUDA device with a D3D10 device in order to achieve maximum interoperability performance.

Parameters:

pD3D10Device - Direct3D device to use for interoperability

device - The CUDA device to use. This device must be among the devices returned when querying **cudaD3D10DeviceListAll** from **cudaD3D10GetDevices**, may be set to −1 to automatically select an appropriate CUDA device.

Returns:
cudaSuccess, **cudaErrorInitializationError**, **cudaErrorInvalidValue**, **cudaErrorSetOnActiveProcess**

Note:
Note that this function may also return error codes from previous, asynchronous launches.

See also:
cudaD3D10GetDevice, **cudaGraphicsD3D10RegisterResource**, **cudaDeviceReset**

cudaError_t cudaD3D10UnmapResources (int count, ID3D10Resource ** ppResources)

Deprecated
This function is deprecated as of CUDA 3.0.

Unmaps the `count` Direct3D resource in `ppResources`.

This function provides the synchronization guarantee that any CUDA kernels issued before **cudaD3D10UnmapResources()** will complete before any Direct3D calls issued after **cudaD3D10UnmapResources()** begin.

If any of `ppResources` have not been registered for use with CUDA or if `ppResources` contains any duplicate entries, then **cudaErrorInvalidResourceHandle** is returned. If any of `ppResources` are not presently mapped for access by CUDA then **cudaErrorUnknown** is returned.

Parameters:
count - Number of resources to unmap for CUDA
ppResources - Resources to unmap for CUDA

Returns:
cudaSuccess, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:
Note that this function may also return error codes from previous, asynchronous launches.

See also:
cudaGraphicsUnmapResources

cudaError_t cudaD3D10UnregisterResource (ID3D10Resource * pResource)

Deprecated
This function is deprecated as of CUDA 3.0.

Unregisters the Direct3D resource `resource` so it is not accessible by CUDA unless registered again.

If `pResource` is not registered, then **cudaErrorInvalidResourceHandle** is returned.

Parameters:
pResource - Resource to unregister

Returns:
cudaSuccess, **cudaErrorInvalidResourceHandle**, **cudaErrorUnknown**

Note:
Note that this function may also return error codes from previous, asynchronous launches.

See also:
cudaGraphicsUnregisterResource

Author

Generated automatically by Doxygen from the source code.