NAME

Graphics Interoperability −

Functions

cudaError_t cudaGraphicsMapResources (int count, **cudaGraphicsResource_t** *resources, **cudaStream_t** stream=0)
Map graphics resources for access by CUDA.

cudaError_t cudaGraphicsResourceGetMappedMipmappedArray (**cudaMipmappedArray_t** *mipmappedArray, **cudaGraphicsResource_t** resource)
Get a mipmapped array through which to access a mapped graphics resource.

cudaError_t cudaGraphicsResourceGetMappedPointer (void **devPtr, size_t *size, **cudaGraphicsResource_t** resource)
Get an device pointer through which to access a mapped graphics resource.

cudaError_t cudaGraphicsResourceSetMapFlags (**cudaGraphicsResource_t** resource, unsigned int flags)
Set usage flags for mapping a graphics resource.

cudaError_t cudaGraphicsSubResourceGetMappedArray (**cudaArray_t** *array, **cudaGraphicsResource_t** resource, unsigned int arrayIndex, unsigned int mipLevel)
Get an array through which to access a subresource of a mapped graphics resource.

cudaError_t cudaGraphicsUnmapResources (int count, **cudaGraphicsResource_t** *resources, **cudaStream_t** stream=0)
Unmap graphics resources.

cudaError_t cudaGraphicsUnregisterResource (**cudaGraphicsResource_t** resource)
Unregisters a graphics resource for access by CUDA.

Detailed Description

\brief graphics interoperability functions of the CUDA runtime API (cuda_runtime_api.h)

This section describes the graphics interoperability functions of the CUDA runtime application programming interface.

Function Documentation

cudaError_t cudaGraphicsMapResources (int count, cudaGraphicsResource_t * resources, cudaStream_t stream = 0)

Maps the `count` graphics resources in `resources` for access by CUDA.

The resources in `resources` may be accessed by CUDA until they are unmapped. The graphics API from which `resources` were registered 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 graphics calls issued before **cudaGraphicsMapResources()** will complete before any subsequent CUDA work issued in `stream` begins.

If `resources` contains any duplicate entries then **cudaErrorInvalidResourceHandle** is returned. If any of `resources` are presently mapped for access by CUDA then **cudaErrorUnknown** is returned.

Parameters:

count - Number of resources to map
resources - Resources to map for CUDA
stream - Stream for synchronization

Returns:

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

Note:

This function uses standard semantics.

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

See also:

cudaGraphicsResourceGetMappedPointer, **cudaGraphicsSubResourceGetMappedArray**, **cudaGraphicsUnmapResources**



cudaError_t cudaGraphicsResourceGetMappedMipmappedArray (cudaMipmappedArray_t * mipmappedArray, cudaGraphicsResource_t resource)

Returns in `*mipmappedArray` a mipmapped array through which the mapped graphics resource `resource` may be accessed. The value set in `mipmappedArray` may change every time that `resource` is mapped.

If `resource` is not a texture then it cannot be accessed via an array and **cudaErrorUnknown** is returned. If `resource` is not mapped then **cudaErrorUnknown** is returned.

Parameters:

mipmappedArray - Returned mipmapped array through which `resource` may be accessed
resource - Mapped resource 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 cudaGraphicsResourceGetMappedPointer (void ** devPtr, size_t * size, cudaGraphicsResource_t resource)

Returns in `*devPtr` a pointer through which the mapped graphics resource `resource` may be accessed. Returns in `*size` the size of the memory in bytes which may be accessed from that pointer. The value set in `devPtr` may change every time that `resource` is mapped.

If `resource` is not a buffer then it cannot be accessed via a pointer and **cudaErrorUnknown** is returned. If `resource` is not mapped then **cudaErrorUnknown** is returned. *

Parameters:

devPtr - Returned pointer through which `resource` may be accessed
size - Returned size of the buffer accessible starting at `*devPtr`
resource - Mapped resource to access

Returns:

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

Note:

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

See also:

cudaGraphicsMapResources, **cudaGraphicsSubResourceGetMappedArray**

cudaError_t cudaGraphicsResourceSetMapFlags (cudaGraphicsResource_t resource, unsigned int flags)

Set `flags` for mapping the graphics resource `resource`.

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

- **cudaGraphicsMapFlagsNone**: Specifies no hints about how `resource` will be used. It is therefore assumed that CUDA may read from or write to `resource`.
- **cudaGraphicsMapFlagsReadOnly**: Specifies that CUDA will not write to `resource`.
- **cudaGraphicsMapFlagsWriteDiscard**: Specifies CUDA will not read from `resource` and will write over the entire contents of `resource`, so none of the data previously stored in `resource` will be preserved.

If `resource` is presently mapped for access by CUDA then **cudaErrorUnknown** is returned. If `flags` is not one of the above values then **cudaErrorInvalidValue** is returned.

Parameters:

resource - 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:
cudaGraphicsMapResources

cudaError_t cudaGraphicsSubResourceGetMappedArray (cudaArray_t * array, cudaGraphicsResource_t resource, unsigned int arrayIndex, unsigned int mipLevel)
Returns in `*array` an array through which the subresource of the mapped graphics resource `resource` which corresponds to array index `arrayIndex` and mipmap level `mipLevel` may be accessed. The value set in `array` may change every time that `resource` is mapped.

If `resource` is not a texture then it cannot be accessed via an array and **cudaErrorUnknown** is returned. If `arrayIndex` is not a valid array index for `resource` then **cudaErrorInvalidValue** is returned. If `mipLevel` is not a valid mipmap level for `resource` then **cudaErrorInvalidValue** is returned. If `resource` is not mapped then **cudaErrorUnknown** is returned.

Parameters:
array - Returned array through which a subresource of `resource` may be accessed
resource - Mapped resource to access
arrayIndex - Array index for array textures or cubemap face index as defined by **cudaGraphicsCubeFace** for cubemap textures for the subresource to access
mipLevel - Mipmap level for the subresource 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 cudaGraphicsUnmapResources (int count, cudaGraphicsResource_t * resources, cudaStream_t stream = 0)
Unmaps the `count` graphics resources in `resources`.

Once unmapped, the resources in `resources` may not be accessed by CUDA until they are mapped again.

This function provides the synchronization guarantee that any CUDA work issued in `stream` before **cudaGraphicsUnmapResources()** will complete before any subsequently issued graphics work begins.

If `resources` contains any duplicate entries then **cudaErrorInvalidResourceHandle** is returned. If any of `resources` are not presently mapped for access by CUDA then **cudaErrorUnknown** is returned.

Parameters:
count - Number of resources to unmap
resources - Resources to unmap
stream - Stream for synchronization

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

Note:
This function uses standard semantics.

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

See also:
cudaGraphicsMapResources



cudaError_t cudaGraphicsUnregisterResource (cudaGraphicsResource_t resource)

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

If `resource` is invalid then **cudaErrorInvalidResourceHandle** is returned.

Parameters:

resource - Resource to unregister

Returns:

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

Note:

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

See also:

cudaGraphicsD3D9RegisterResource, **cudaGraphicsD3D10RegisterResource**, **cudaGraphicsD3D11RegisterResource**, **cudaGraphicsGLRegisterBuffer**, **cudaGraphicsGLRegisterImage**

Author

Generated automatically by Doxygen from the source code.

