NAME

Module Management −

Functions

CUresult cuLinkAddData (CUlinkState state, **CUjitInputType** type, void *data, size_t size, const char *name, unsigned int numOptions, **CUjit_option** *options, void **optionValues)
Add an input to a pending linker invocation.

CUresult cuLinkAddFile (CUlinkState state, **CUjitInputType** type, const char *path, unsigned int numOptions, **CUjit_option** *options, void **optionValues)
Add a file input to a pending linker invocation.

CUresult cuLinkComplete (CUlinkState state, void **cubinOut, size_t *sizeOut)
Complete a pending linker invocation.

CUresult cuLinkCreate (unsigned int numOptions, **CUjit_option** *options, void **optionValues, CUlinkState *stateOut)
Creates a pending JIT linker invocation.

CUresult cuLinkDestroy (CUlinkState state)
Destroys state for a JIT linker invocation.

CUresult cuModuleGetFunction (**CUfunction** *hfunc, **CUmodule** hmod, const char *name)
Returns a function handle.

CUresult cuModuleGetGlobal (**CUdeviceptr** *dptr, size_t *bytes, **CUmodule** hmod, const char *name)
Returns a global pointer from a module.

CUresult cuModuleGetSurfRef (**CUsurfref** *pSurfRef, **CUmodule** hmod, const char *name)
Returns a handle to a surface reference.

CUresult cuModuleGetTexRef (**CUtexref** *pTexRef, **CUmodule** hmod, const char *name)
Returns a handle to a texture reference.

CUresult cuModuleLoad (**CUmodule** *module, const char *fname)
Loads a compute module.

CUresult cuModuleLoadData (**CUmodule** *module, const void *image)
Load a module's data.

CUresult cuModuleLoadDataEx (**CUmodule** *module, const void *image, unsigned int numOptions, **CUjit_option** *options, void **optionValues)
Load a module's data with options.

CUresult cuModuleLoadFatBinary (**CUmodule** *module, const void *fatCubin)
Load a module's data.

CUresult cuModuleUnload (**CUmodule** hmod)
Unloads a module.

Detailed Description

\brief module management functions of the low-level CUDA driver API (**cuda.h**)

This section describes the module management functions of the low-level CUDA driver application programming interface.

Function Documentation

CUresult cuLinkAddData (CUlinkState state, CUjitInputType type, void * data, size_t size, const char * name, unsigned int numOptions, CUjit_option * options, void ** optionValues)

Ownership of `data` is retained by the caller. No reference is retained to any inputs after this call returns.

This method accepts only compiler options, which are used if the data must be compiled from PTX, and does not accept any of **CU_JIT_WALL_TIME**, **CU_JIT_INFO_LOG_BUFFER**, **CU_JIT_ERROR_LOG_BUFFER**, **CU_JIT_TARGET_FROM_CUCONTEXT**, or **CU_JIT_TARGET**.

Parameters:

state A pending linker action.
type The type of the input data.
data The input data. PTX must be NULL-terminated.
size The length of the input data.
name An optional name for this input in log messages.
numOptions Size of options.



options Options to be applied only for this input (overrides options from **cuLinkCreate**).
optionValues Array of option values, each cast to void *.

Returns:
CUDA_SUCCESS, **CUDA_ERROR_INVALID_HANDLE**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_IMAGE**, **CUDA_ERROR_INVALID_PTX**, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_NO_BINARY_FOR_GPU**

See also:
cuLinkCreate, **cuLinkAddFile**, **cuLinkComplete**, **cuLinkDestroy**

CUresult cuLinkAddFile (CUlinkState state, CUjitInputType type, const char * path, unsigned int numOptions, CUjit_option * options, void ** optionValues)
No reference is retained to any inputs after this call returns.

This method accepts only compiler options, which are used if the input must be compiled from PTX, and does not accept any of **CU_JIT_WALL_TIME**, **CU_JIT_INFO_LOG_BUFFER**, **CU_JIT_ERROR_LOG_BUFFER**, **CU_JIT_TARGET_FROM_CUCONTEXT**, or **CU_JIT_TARGET**.

This method is equivalent to invoking **cuLinkAddData** on the contents of the file.

Parameters:
state A pending linker action
type The type of the input data
path Path to the input file
numOptions Size of options
options Options to be applied only for this input (overrides options from **cuLinkCreate**)
optionValues Array of option values, each cast to void *

Returns:
CUDA_SUCCESS, **CUDA_ERROR_FILE_NOT_FOUND** **CUDA_ERROR_INVALID_HANDLE**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_IMAGE**, **CUDA_ERROR_INVALID_PTX**, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_NO_BINARY_FOR_GPU**

See also:
cuLinkCreate, **cuLinkAddData**, **cuLinkComplete**, **cuLinkDestroy**

CUresult cuLinkComplete (CUlinkState state, void ** cubinOut, size_t * sizeOut)
Completes the pending linker action and returns the cubin image for the linked device code, which can be used with **cuModuleLoadData**. The cubin is owned by `state`, so it should be loaded before `state` is destroyed via **cuLinkDestroy**. This call does not destroy `state`.

Parameters:
state A pending linker invocation
cubinOut On success, this will point to the output image
sizeOut Optional parameter to receive the size of the generated image

Returns:
CUDA_SUCCESS, **CUDA_ERROR_INVALID_HANDLE**, **CUDA_ERROR_OUT_OF_MEMORY**

See also:
cuLinkCreate, **cuLinkAddData**, **cuLinkAddFile**, **cuLinkDestroy**, **cuModuleLoadData**

CUresult cuLinkCreate (unsigned int numOptions, CUjit_option * options, void ** optionValues, CUlinkState * stateOut)
If the call is successful, the caller owns the returned CUlinkState, which should eventually be destroyed with **cuLinkDestroy**. The device code machine size (32 or 64 bit) will match the calling application.

Both linker and compiler options may be specified. Compiler options will be applied to inputs to this linker action which must be compiled from PTX. The options **CU_JIT_WALL_TIME**, **CU_JIT_INFO_LOG_BUFFER_SIZE_BYTES**, and **CU_JIT_ERROR_LOG_BUFFER_SIZE_BYTES** will accumulate data until the CUlinkState is destroyed.



`optionValues` must remain valid for the life of the CUlinkState if output options are used. No other references to inputs are maintained after this call returns.

Parameters:

numOptions Size of options arrays
options Array of linker and compiler options
optionValues Array of option values, each cast to void *
stateOut On success, this will contain a CUlinkState to specify and complete this action

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_OUT_OF_MEMORY**

Note:

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

See also:

cuLinkAddData, **cuLinkAddFile**, **cuLinkComplete**, **cuLinkDestroy**

CUresult cuLinkDestroy (CUlinkState state)

Parameters:

state State object for the linker invocation

Returns:

CUDA_SUCCESS, **CUDA_ERROR_INVALID_HANDLE**

See also:

cuLinkCreate

CUresult cuModuleGetFunction (CUfunction * hfunc, CUmodule hmod, const char * name)

Returns in `*hfunc` the handle of the function of name `name` located in module `hmod`. If no function of that name exists, **cuModuleGetFunction()** returns **CUDA_ERROR_NOT_FOUND**.

Parameters:

hfunc - Returned function handle
hmod - Module to retrieve function from
name - Name of function to retrieve

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_NOT_FOUND**

Note:

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

See also:

cuModuleGetGlobal, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleGetGlobal (CUdeviceptr * dptr, size_t * bytes, CUmodule hmod, const char * name)

Returns in `*dptr` and `*bytes` the base pointer and size of the global of name `name` located in module `hmod`. If no variable of that name exists, **cuModuleGetGlobal()** returns **CUDA_ERROR_NOT_FOUND**. Both parameters `dptr` and `bytes` are optional. If one of them is NULL, it is ignored.

Parameters:

dptr - Returned global device pointer
bytes - Returned global size in bytes
hmod - Module to retrieve global from
name - Name of global to retrieve

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_NOT_FOUND**



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

See also:
cuModuleGetFunction, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleGetSurfRef (CUsurfref * pSurfRef, CUmodule hmod, const char * name)
Returns in `*pSurfRef` the handle of the surface reference of name `name` in the module `hmod`. If no surface reference of that name exists, **cuModuleGetSurfRef()** returns **CUDA_ERROR_NOT_FOUND**.

Parameters:
pSurfRef - Returned surface reference
hmod - Module to retrieve surface reference from
name - Name of surface reference to retrieve

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_NOT_FOUND**

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

See also:
cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleGetTexRef (CUtexref * pTexRef, CUmodule hmod, const char * name)
Returns in `*pTexRef` the handle of the texture reference of name `name` in the module `hmod`. If no texture reference of that name exists, **cuModuleGetTexRef()** returns **CUDA_ERROR_NOT_FOUND**. This texture reference handle should not be destroyed, since it will be destroyed when the module is unloaded.

Parameters:
pTexRef - Returned texture reference
hmod - Module to retrieve texture reference from
name - Name of texture reference to retrieve

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_NOT_FOUND**

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

See also:
cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetSurfRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleLoad (CUmodule * module, const char * fname)
Takes a filename `fname` and loads the corresponding module `module` into the current context. The CUDA driver API does not attempt to lazily allocate the resources needed by a module; if the memory for functions and data (constant and global) needed by the module cannot be allocated, **cuModuleLoad()** fails. The file should be a *cubin* file as output by **nvcc**, or a *PTX* file either as output by **nvcc** or handwritten, or a *fatbin* file as output by **nvcc** from toolchain 4.0 or later.

Parameters:
module - Returned module
fname - Filename of module to load

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_PTX**,



CUDA_ERROR_NOT_FOUND, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_FILE_NOT_FOUND**, **CUDA_ERROR_NO_BINARY_FOR_GPU**, **CUDA_ERROR_SHARED_OBJECT_SYMBOL_NOT_FOUND**, **CUDA_ERROR_SHARED_OBJECT_INIT_FAILED**

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

See also:
cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetTexRef**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleLoadData (CUmodule * module, const void * image)
Takes a pointer `image` and loads the corresponding module `module` into the current context. The pointer may be obtained by mapping a *cubin* or *PTX* or *fatbin* file, passing a *cubin* or *PTX* or *fatbin* file as a NULL-terminated text string, or incorporating a *cubin* or *fatbin* object into the executable resources and using operating system calls such as Windows `FindResource()` to obtain the pointer.

Parameters:
module - Returned module
image - Module data to load

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_PTX**, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_NO_BINARY_FOR_GPU**, **CUDA_ERROR_SHARED_OBJECT_SYMBOL_NOT_FOUND**, **CUDA_ERROR_SHARED_OBJECT_INIT_FAILED**

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

See also:
cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleLoadDataEx (CUmodule * module, const void * image, unsigned int numOptions, CUjit_option * options, void ** optionValues)
Takes a pointer `image` and loads the corresponding module `module` into the current context. The pointer may be obtained by mapping a *cubin* or *PTX* or *fatbin* file, passing a *cubin* or *PTX* or *fatbin* file as a NULL-terminated text string, or incorporating a *cubin* or *fatbin* object into the executable resources and using operating system calls such as Windows `FindResource()` to obtain the pointer. Options are passed as an array via `options` and any corresponding parameters are passed in `optionValues`. The number of total options is supplied via `numOptions`. Any outputs will be returned via `optionValues`.

Parameters:
module - Returned module
image - Module data to load
numOptions - Number of options
options - Options for JIT
optionValues - Option values for JIT

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_PTX**, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_NO_BINARY_FOR_GPU**, **CUDA_ERROR_SHARED_OBJECT_SYMBOL_NOT_FOUND**, **CUDA_ERROR_SHARED_OBJECT_INIT_FAILED**

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

See also:



cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadFatBinary**, **cuModuleUnload**

CUresult cuModuleLoadFatBinary (CUmodule * module, const void * fatCubin)

Takes a pointer `fatCubin` and loads the corresponding module `module` into the current context. The pointer represents a *fat binary* object, which is a collection of different *cubin* and/or *PTX* files, all representing the same device code, but compiled and optimized for different architectures.

Prior to CUDA 4.0, there was no documented API for constructing and using fat binary objects by programmers. Starting with CUDA 4.0, fat binary objects can be constructed by providing the *-fatbin option* to **nvcc**. More information can be found in the **nvcc** document.

Parameters:

module - Returned module
fatCubin - Fat binary to load

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_PTX**, **CUDA_ERROR_NOT_FOUND**, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_NO_BINARY_FOR_GPU**, **CUDA_ERROR_SHARED_OBJECT_SYMBOL_NOT_FOUND**, **CUDA_ERROR_SHARED_OBJECT_INIT_FAILED**

Note:

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

See also:

cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleUnload**

CUresult cuModuleUnload (CUmodule hmod)

Unloads a module `hmod` from the current context.

Parameters:

hmod - Module to unload

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**

Note:

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

See also:

cuModuleGetFunction, **cuModuleGetGlobal**, **cuModuleGetTexRef**, **cuModuleLoad**, **cuModuleLoadData**, **cuModuleLoadDataEx**, **cuModuleLoadFatBinary**

Author

Generated automatically by Doxygen from the source code.

