cuDLA API

API Reference Manual

v12.4 | March 2024

Table of Contents

Chapter 1. Modules

Here is a list of all modules:

- Data types used by cuDLA driver
- cuDLA API

1.1. Data types used by cuDLA driver

union cudlaDevAttribute

struct cudlaExternalMemoryHandleDesc_t

struct cudlaExternalSemaphoreHandleDesc_t

struct CudlaFence

union cudlaModuleAttribute

struct cudlaModuleTensorDescriptor

struct cudlaSignalEvents

struct cudlaTask

struct cudlaWaitEvents

enum cudlaAccessPermissionFlags

Access permission flags for importing NvSciBuffers

Values

CUDLA_READ_WRITE_PERM = 0
Flag to import memory with read-write permission

CUDLA_READ_ONLY_PERM = 1
Flag to import memory with read-only permission

CUDLA_TASK_STATISTICS = 1<<1
Flag to indicate buffer as layerwise statistics buffer.

enum cudlaDevAttributeType

Device attribute type.

Values

CUDLA_UNIFIED_ADDRESSING = 0
Flag to check for support for UVA.

CUDLA_DEVICE_VERSION = 1

Flag to check for DLA HW version.

enum cudlaFenceType

Supported fence types.

Values

CUDLA_NVSCISYNC_FENCE = 1
NvSciSync fence type for EOF.
CUDLA_NVSCISYNC_FENCE_SOF = 2

enum cudlaMode

Device creation modes.

Values

CUDLA_CUDA_DLA = 0
Hyrbid mode.
CUDLA_STANDALONE = 1
Standalone mode.

enum cudlaModuleAttributeType

Module attribute types.

Values

CUDLA_NUM_INPUT_TENSORS = 0
Flag to retrieve number of input tensors.
CUDLA_NUM_OUTPUT_TENSORS = 1
Flag to retrieve number of output tensors.
CUDLA_INPUT_TENSOR_DESCRIPTORS = 2
Flag to retrieve all the input tensor descriptors.
CUDLA_OUTPUT_TENSOR_DESCRIPTORS = 3
Flag to retrieve all the output tensor descriptors.
CUDLA_NUM_OUTPUT_TASK_STATISTICS = 4
Flag to retrieve total number of output task statistics buffer.
CUDLA_OUTPUT_TASK_STATISTICS_DESCRIPTORS = 5
Flag to retrieve all the output task statistics descriptors.

enum cudlaModuleLoadFlags

Module load flags for cudlaModuleLoadFromMemory.

Values

CUDLA_MODULE_DEFAULT = 0
Default flag.

CUDLA_MODULE_ENABLE_FAULT_DIAGNOSTICS = 1
Flag to load a module that is used to perform permanent fault diagnostics for DLA HW.

enum cudlaNvSciSyncAttributes

cuDLA NvSciSync attributes.

Values

CUDLA_NVSCISYNC_ATTR_WAIT = 1
Wait attribute.

CUDLA_NVSCISYNC_ATTR_SIGNAL = 2
Signal attribute.

enum cudlaStatus

Error codes.

Values

cudlaSuccess = 0
The API call returned with no errors.

cudlaErrorInvalidParam = 1
This indicates that one or more parameters passed to the API is/are incorrect.

cudlaErrorOutOfResources = 2
This indicates that the API call failed due to lack of underlying resources.

cudlaErrorCreationFailed = 3
This indicates that an internal error occurred during creation of device handle.

cudlaErrorInvalidAddress = 4
This indicates that the memory object being passed in the API call has not been registered before.

cudlaErrorOs = 5
This indicates that an OS error occurred.

cudlaErrorCuda = 6
This indicates that there was an error in a CUDA operation as part of the API call.

cudlaErrorUmd = 7
This indicates that there was an error in the DLA runtime for the API call.

cudlaErrorInvalidDevice = 8
This indicates that the device handle passed to the API call is invalid.

cudlaErrorInvalidAttribute = 9
This indicates that an invalid attribute is being requested.

cudlaErrorIncompatibleDlaSWVersion = 10
This indicates that the underlying DLA runtime is incompatible with the current cuDLA version.
cudlaErrorMemoryRegistered = 11
This indicates that the memory object is already registered.
cudlaErrorInvalidModule = 12
This indicates that the module being passed is invalid.
cudlaErrorUnsupportedOperation = 13
This indicates that the operation being requested by the API call is unsupported.
cudlaErrorNvSci = 14
This indicates that the NvSci operation requested by the API call failed.
cudlaErrorDlaErrInvalidInput = 0x40000001
DLA HW Error.
cudlaErrorDlaErrInvalidPreAction = 0x40000002
DLA HW Error.
cudlaErrorDlaErrNoMem = 0x40000003
DLA HW Error.
cudlaErrorDlaErrProcessorBusy = 0x40000004
DLA HW Error.
cudlaErrorDlaErrTaskStatusMismatch = 0x40000005
DLA HW Error.
cudlaErrorDlaErrEngineTimeout = 0x40000006
DLA HW Error.
cudlaErrorDlaErrDataMismatch = 0x40000007
DLA HW Error.
cudlaErrorUnknown = 0x7fffffff
This indicates that an unknown error has occurred.

enum cudlaSubmissionFlags

Task submission flags for cudlaSubmitTask.

Values

CUDLA_SUBMIT_NOOP = 1
Flag to specify that the submitted task must be bypassed for execution.
CUDLA_SUBMIT_SKIP_LOCK_ACQUIRE = 1<<1
Flag to specify that the global lock acquire must be skipped.
CUDLA_SUBMIT_DIAGNOSTICS_TASK = 1<<2
Flag to specify that the submitted task is to run permanent fault diagnostics for DLA HW.

typedef cudlaDevHandle_t *cudlaDevHandle

cuDLA Device Handle

typedef cudlaModule_t *cudlaModule

cuDLA Module Handle

1.2. Modules

Here is a list of all modules:

- Data types used by cuDLA driver
- cuDLA API

1.1. Data types used by cuDLA driver

union cudlaDevAttribute

struct cudlaExternalMemoryHandleDesc_t

struct cudlaExternalSemaphoreHandleDesc_t

struct CudlaFence

union cudlaModuleAttribute

struct cudlaModuleTensorDescriptor

struct cudlaSignalEvents

struct cudlaTask

struct cudlaWaitEvents

enum cudlaAccessPermissionFlags

Access permission flags for importing NvSciBuffers

Values

CUDLA_READ_WRITE_PERM = 0
Flag to import memory with read-write permission

CUDLA_READ_ONLY_PERM = 1

Flag to import memory with read-only permission

CUDLA_TASK_STATISTICS = 1<<1

Flag to indicate buffer as layerwise statistics buffer.

enum cudlaDevAttributeType

Device attribute type.

Values

CUDLA_UNIFIED_ADDRESSING = 0

Flag to check for support for UVA.

CUDLA_DEVICE_VERSION = 1

Flag to check for DLA HW version.

enum cudlaFenceType

Supported fence types.

Values

CUDLA_NVSCISYNC_FENCE = 1

NvSciSync fence type for EOF.

CUDLA_NVSCISYNC_FENCE_SOF = 2

enum cudlaMode

Device creation modes.

Values

CUDLA_CUDA_DLA = 0

Hyrbid mode.

CUDLA_STANDALONE = 1

Standalone mode.

enum cudlaModuleAttributeType

Module attribute types.

Values

CUDLA_NUM_INPUT_TENSORS = 0

Flag to retrieve number of input tensors.

CUDLA_NUM_OUTPUT_TENSORS = 1

Flag to retrieve number of output tensors.

CUDLA_INPUT_TENSOR_DESCRIPTORS = 2

Flag to retrieve all the input tensor descriptors.

CUDLA_OUTPUT_TENSOR_DESCRIPTORS = 3
Flag to retrieve all the output tensor descriptors.
CUDLA_NUM_OUTPUT_TASK_STATISTICS = 4
Flag to retrieve total number of output task statistics buffer.
CUDLA_OUTPUT_TASK_STATISTICS_DESCRIPTORS = 5
Flag to retrieve all the output task statistics descriptors.

enum cudlaModuleLoadFlags

Module load flags for cudlaModuleLoadFromMemory.

Values

CUDLA_MODULE_DEFAULT = 0
Default flag.
CUDLA_MODULE_ENABLE_FAULT_DIAGNOSTICS = 1
Flag to load a module that is used to perform permanent fault diagnostics for DLA HW.

enum cudlaNvSciSyncAttributes

cuDLA NvSciSync attributes.

Values

CUDLA_NVSCISYNC_ATTR_WAIT = 1
Wait attribute.
CUDLA_NVSCISYNC_ATTR_SIGNAL = 2
Signal attribute.

enum cudlaStatus

Error codes.

Values

cudlaSuccess = 0
The API call returned with no errors.
cudlaErrorInvalidParam = 1
This indicates that one or more parameters passed to the API is/are incorrect.
cudlaErrorOutOfResources = 2
This indicates that the API call failed due to lack of underlying resources.
cudlaErrorCreationFailed = 3
This indicates that an internal error occurred during creation of device handle.
cudlaErrorInvalidAddress = 4
This indicates that the memory object being passed in the API call has not been registered before.
cudlaErrorOs = 5

This indicates that an OS error occurred.

cudlaErrorCuda = 6

This indicates that there was an error in a CUDA operation as part of the API call.

cudlaErrorUmd = 7

This indicates that there was an error in the DLA runtime for the API call.

cudlaErrorInvalidDevice = 8

This indicates that the device handle passed to the API call is invalid.

cudlaErrorInvalidAttribute = 9

This indicates that an invalid attribute is being requested.

cudlaErrorIncompatibleDlaSWVersion = 10

This indicates that the underlying DLA runtime is incompatible with the current cuDLA version.

cudlaErrorMemoryRegistered = 11

This indicates that the memory object is already registered.

cudlaErrorInvalidModule = 12

This indicates that the module being passed is invalid.

cudlaErrorUnsupportedOperation = 13

This indicates that the operation being requested by the API call is unsupported.

cudlaErrorNvSci = 14

This indicates that the NvSci operation requested by the API call failed.

cudlaErrorDlaErrInvalidInput = 0x40000001

DLA HW Error.

cudlaErrorDlaErrInvalidPreAction = 0x40000002

DLA HW Error.

cudlaErrorDlaErrNoMem = 0x40000003

DLA HW Error.

cudlaErrorDlaErrProcessorBusy = 0x40000004

DLA HW Error.

cudlaErrorDlaErrTaskStatusMismatch = 0x40000005

DLA HW Error.

cudlaErrorDlaErrEngineTimeout = 0x40000006

DLA HW Error.

cudlaErrorDlaErrDataMismatch = 0x40000007

DLA HW Error.

cudlaErrorUnknown = 0x7fffffff

This indicates that an unknown error has occurred.

enum cudlaSubmissionFlags

Task submission flags for cudlaSubmitTask.

Values

CUDLA_SUBMIT_NOOP = 1

Flag to specify that the submitted task must be bypassed for execution.

CUDLA_SUBMIT_SKIP_LOCK_ACQUIRE = 1<<1

Flag to specify that the global lock acquire must be skipped.

CUDLA_SUBMIT_DIAGNOSTICS_TASK = 1<<2

Flag to specify that the submitted task is to run permanent fault diagnostics for DLA HW.

typedef cudlaDevHandle_t *cudlaDevHandle

cuDLA Device Handle

typedef cudlaModule_t *cudlaModule

cuDLA Module Handle

Chapter 2. Data Structures

Here are the data structures with brief descriptions:

2.1. cudlaDevAttribute Union Reference

Device attribute.

uint32_t cudlaDevAttribute::deviceVersion

DLA device version. Xavier has 1.0 and Orin has 2.0.

uint8_t cudlaDevAttribute::unifiedAddressingSupported

Returns 0 if unified addressing is not supported.

2.2. cudlaExternalMemoryHandleDesc_t Struct Reference

External memory handle descriptor.

const void *cudlaExternalMemoryHandleDesc_t::extBufObject

A handle representing an external memory object.

unsigned long long cudlaExternalMemoryHandleDesc_t::size

Size of the memory allocation

2.3. cudlaExternalSemaphoreHandleDesc_t Struct Reference

External semaphore handle descriptor.

const void *cudlaExternalSemaphoreHandleDesc_t::extSyncObject

A handle representing an external synchronization object.

2.4. CudlaFence Struct Reference

Fence description.

void *CudlaFence::fence

Fence.

cudlaFenceType CudlaFence::type

Fence type.

2.5. cudlaModuleAttribute Union Reference

Module attribute.

cudlaModuleTensorDescriptor *cudlaModuleAttribute::inputTensorDesc

Returns an array of input tensor descriptors.

uint32_t cudlaModuleAttribute::numInputTensors

Returns the number of input tensors.

uint32_t cudlaModuleAttribute::numOutputTensors

Returns the number of output tensors.

cudlaModuleTensorDescriptor *cudlaModuleAttribute::outputTensorDesc

Returns an array of output tensor descriptors.

2.6. cudlaModuleTensorDescriptor Struct Reference

Tensor descriptor.

2.7. cudlaSignalEvents Struct Reference

Signal events for cudlaSubmitTask

const **cudlaSignalEvents::devPtrs

Array of registered synchronization objects (via cudlaImportExternalSemaphore).

CudlaFence *cudlaSignalEvents::eofFences

Array of fences pointers for all the signal events corresponding to the synchronization objects.

uint32_t cudlaSignalEvents::numEvents

Total number of signal events.

2.8. cudlaTask Struct Reference

Structure of Task.

const **cudlaTask::inputTensor

Array of input tensors.

cudlaModule cudlaTask::moduleHandle

cuDLA module handle.

uint32_t cudlaTask::numInputTensors

Number of input tensors.

uint32_t cudlaTask::numOutputTensors

Number of output tensors.

const **cudlaTask::outputTensor

Array of output tensors.

cudlaSignalEvents *cudlaTask::signalEvents

Signal events.

const cudlaWaitEvents *cudlaTask::waitEvents

Wait events.

2.9. cudlaWaitEvents Struct Reference

Wait events for cudlaSubmitTask.

uint32_t cudlaWaitEvents::numEvents

Total number of wait events.

const CudlaFence *cudlaWaitEvents::preFences

Array of fence pointers for all the wait events.

Chapter 3. Data Fields

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

deviceVersion
- cudlaDevAttribute

devPtrs
- cudlaSignalEvents

eofFences
- cudlaSignalEvents

extBufObject
- cudlaExternalMemoryHandleDesc

extSyncObject
- cudlaExternalSemaphoreHandleDesc

fence
- CudlaFence

inputTensor
- cudlaTask

inputTensorDesc
- cudlaModuleAttribute

moduleHandle
- cudlaTask

numEvents
- cudlaWaitEvents
- cudlaSignalEvents

numInputTensors
- cudlaTask
- cudlaModuleAttribute

numOutputTensors
- cudlaTask
- cudlaModuleAttribute

outputTensor
- cudlaTask

outputTensorDesc
- cudlaModuleAttribute

preFences
cudlaWaitEvents
signalEvents
cudlaTask
size
cudlaExternalMemoryHandleDesc
type
CudlaFence
unifiedAddressingSupported
cudlaDevAttribute
waitEvents
cudlaTask

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com