CUDA DEBUGGER API

TRM-06710-001 _v5.5 | July 2013

API Reference Manual

TABLE OF CONTENTS

Chapter 1. INTRODUCTION

This document describes the API for the set routines and data structures available in the CUDA library to any debugger.

Starting with 3.0, the CUDA debugger API includes several major changes, of which only few are directly visible to end-users:

- Performance is greatly improved, both with respect to interactions with the debugger and the performance of applications being debugged.
- The format of cubins has changed to ELF and, as a consequence, most restrictions on debug compilations have been lifted. More information about the new object format is included below.

The debugger API has significantly changed, reflected in the CUDA-GDB sources.

1.1. Debugger API

The CUDA Debugger API was developed with the goal of adhering to the following principles:

- Policy free
- Explicit
- Axiomatic
- Extensible
- Machine oriented

Being explicit is another way of saying that we minimize the assumptions we make. As much as possible the API reflects machine state, not internal state.

There are two major "modes" of the devices: stopped or running. We switch between these modes explicitly with suspendDevice and resumeDevice, though the machine may suspend on its own accord, for example when hitting a breakpoint.

Only when stopped, can we query the machine's state. Warp state includes which function is it runnning, which block, which lanes are valid, etc.

1.2. ELF and DWARF

CUDA applications are compiled in ELF binary format.

DWARF device information is obtained through a CUDBGEvent of type CUDBG_EVENT_ELF_IMAGE_LOADED. This means that the information is not available until runtime, after the CUDA driver has loaded.

DWARF device information contains physical addresses for all device memory regions except for code memory. The address class field (DW_AT_address_class) is set for all device variables, and is used to indicate the memory segment type (ptxStorageKind). The physical addresses must be accessed using several segment-specific API calls.

For memory reads, see:

- CUDBGAPI_st::readCodeMemory()
- CUDBGAPI_st::readConstMemory()
- CUDBGAPI_st::readGlobalMemory()
- CUDBGAPI_st::readParamMemory()
- CUDBGAPI_st::readSharedMemory()
- CUDBGAPI_st::readLocalMemory()
- CUDBGAPI_st::readTextureMemory()

For memory writes, see:

- CUDBGAPI_st::writeGlobalMemory()
- CUDBGAPI_st::writeParamMemory()
- CUDBGAPI_st::writeSharedMemory()
- CUDBGAPI_st::writeLocalMemory()

Access to code memory requires a virtual address. This virtual address is embedded for all device code sections in the device ELF image. See the API call:

- CUDBGAPI_st::readVirtualPC()

Here is a typical DWARF entry for a device variable located in memory:

```
<2><321>: Abbrev Number: 18 (DW_TAG_formal_parameter)
     DW_AT_decl_file   : 27
     DW_AT_decl_line   : 5
     DW_AT_name        : res
     DW_AT_type        : <2c6>
     DW_AT_location    : 9 byte block: 3 18 0 0 0 0 0 0 0       (DW_OP_addr: 18)
     DW_AT_address_class: 7
```

The above shows that variable 'res' has an address class of 7 (ptxParamStorage). Its location information shows it is located at address 18 within the parameter memory segment.

Local variables are no longer spilled to local memory by default. The DWARF now contains variable-to-register mapping and liveness information for all variables. It can be the case that variables are spilled to local memory, and this is all contained in the DWARF information which is ULEB128 encoded (as a DW_OP_regx stack operation in the DW_AT_location attribute).

Here is a typical DWARF entry for a variable located in a local register:

```
<3><359>: Abbrev Number: 20 (DW_TAG_variable)
     DW_AT_decl_file   : 27
     DW_AT_decl_line   : 7
     DW_AT_name        : c
     DW_AT_type        : <1aa>
     DW_AT_location    : 7 byte block: 90 b9 e2 90 b3 d6 4      (DW_OP_regx:
 160631632185)
     DW_AT_address_class: 2
```

This shows variable 'c' has address class 2 (ptxRegStorage) and its location can be found by decoding the ULEB128 value, DW_OP_regx: 160631632185. See cuda-tdep.c in the cuda-gdb source drop for information on decoding this value and how to obtain which physical register holds this variable during a specific device PC range.

Access to physical registers liveness information requires a 0-based physical PC. See the API call:

- CUDBGAPI_st::readPC()

1.3. ABI Support

ABI support is handled through the following thread API calls:

- CUDBGAPI_st::readCallDepth()
- CUDBGAPI_st::readReturnAddress()
- CUDBGAPI_st::readVirtualReturnAddress()

The return address is not accessible on the local stack and the API call must be used to access its value.

For more information, please refer to the ABI documentation titled "Fermi ABI: Application Binary Interface".

1.4. Exception Reporting

Some kernel exceptions are reported as device events and accessible via the API call:

- CUDBGAPI_st::readLaneException()

The reported exceptions are listed in the CUDBGException_t enum type. Each prefix, (Device, Warp, Lane), refers to the precision of the exception. That is, the lowest known execution unit that is responsible for the origin of the exception. All lane errors are precise; the exact instruction and lane that caused the error are known. Warp errors are typically within a few instructions of where the actual error occurred, but the exact lane within the warp is not known. On device errors, we *may* know the *kernel* that caused it. Explanations about each exception type can be found in the documentation of the struct.

Exception reporting is only supported on Fermi (sm_20 or greater).

1.5. Attaching and Detaching

The debug client must take the following steps to attach to a running CUDA application:

1. Attach to the CPU process corresponding to the CUDA application. The CPU part of the application will be frozen at this point.
2. Check to see if the CUDBG_IPC_FLAG_NAME variable is accessible from the memory space of the application. If not, it implies that the application has not loaded the CUDA driver, and the attaching to the application is complete.
3. Make a dynamic function call to the function cudbgApiInit() with an argument of "2", i.e., "cudbgApiInit(2)". This causes a helper process to be forked off from the application, which assists in attaching to the CUDA process.
4. Ensure that the initialization of the CUDA debug API is complete, or wait till API initialization is successful.
5. Make the "initializeAttachStub()" API call to initialize the helper process that was forked off from the application earlier.
6. Read the value of the CUDBG_ATTACH_HANDLER_AVAILABLE variable from the memory space of the application:
   - If the value is non-zero, resume the CUDA application so that more data can be collected about the application and sent to the debugger. When the application is resumed, the debug client can expect to receive various CUDA events from the CUDA application. Once all state has been collected, the debug client will receive the event CUDBG_EVENT_ATTACH_COMPLETE.
   - If the value is zero, there is no more attach data to collect. Set the CUDBG_IPC_FLAG_NAME variable to 1 in the application's process space, which enables further events from the CUDA application.
7. At this point, attaching to the CUDA application is complete and all GPUs belonging to the CUDA application will be suspended.

The debug client must take the following steps to detach from a running CUDA application:

1. Check to see if the CUDBG_IPC_FLAG_NAME variable is accessible from the memory space of the application, and that the CUDA debug API is initialized. If

either of these conditions is not met, treat the application as CPU-only and detach from the application.

2. Next, make the "clearAttachState" API call to prepare the CUDA application for detach.
3. Read the value of the CUDBG_ATTACH_HANDLER_AVAILABLE variable from the memory space of the application. If the value is non-zero, make the "requestCleanupOnDetach" API call.
4. Set the CUDBG_DEBUGGER_INITIALIZED variable to 0 in the memory space of the application. This makes sure the debugger is reinitialized from scratch if the debug client re-attaches to the application in the future.
5. If the value of the CUDBG_ATTACH_HANDLER_AVAILABLE variable was found to be non-zero in step 3, delete all breakpoints and resume the CUDA application. This allows the CUDA driver to perform cleanups before the debug client detaches from it. Once the cleanup is complete, the debug client will receive the event CUDBG_EVENT_DETACH_COMPLETE.
6. Set the CUDBG_IPC_FLAG_NAME variable to zero in the memory space of the application. This prevents any more callbacks from the CUDA application to the debugger.
7. The client must then finalize the CUDA debug API.
8. Finally, detach from the CPU part of the CUDA application. At this point all GPUs belonging to the CUDA application will be resumed.

Chapter 2. MODULES

Here is a list of all modules:

- General
- Initialization
- Device Execution Control
- Breakpoints
- Device State Inspection
- Device State Alteration
- Grid Properties
- Device Properties
- DWARF Utilities
- Events

2.1. General

enum CUDBGResult

Result values of all the API routines.

Values

CUDBG_SUCCESS = 0x0000
The API call executed successfully.

CUDBG_ERROR_UNKNOWN = 0x0001
Error type not listed below.

CUDBG_ERROR_BUFFER_TOO_SMALL = 0x0002
Cannot copy all the queried data into the buffer argument.

CUDBG_ERROR_UNKNOWN_FUNCTION = 0x0003
Function cannot be found in the CUDA kernel.

CUDBG_ERROR_INVALID_ARGS = 0x0004
Wrong use of arguments (NULL pointer, illegal value,....).
CUDBG_ERROR_UNINITIALIZED = 0x0005
Debugger API has not yet been properly initialized.
CUDBG_ERROR_INVALID_COORDINATES = 0x0006
Invalid block or thread coordinates were provided.
CUDBG_ERROR_INVALID_MEMORY_SEGMENT = 0x0007
Invalid memory segment requested.
CUDBG_ERROR_INVALID_MEMORY_ACCESS = 0x0008
Requested address (+size) is not within proper segment boundaries.
CUDBG_ERROR_MEMORY_MAPPING_FAILED = 0x0009
Memory is not mapped and cannot be mapped.
CUDBG_ERROR_INTERNAL = 0x000a
A debugger internal error occurred.
CUDBG_ERROR_INVALID_DEVICE = 0x000b
Specified device cannot be found.
CUDBG_ERROR_INVALID_SM = 0x000c
Specified sm cannot be found.
CUDBG_ERROR_INVALID_WARP = 0x000d
Specified warp cannot be found.
CUDBG_ERROR_INVALID_LANE = 0x000e
Specified lane cannot be found.
CUDBG_ERROR_SUSPENDED_DEVICE = 0x000f
The requested operation is not allowed when the device is suspended.
CUDBG_ERROR_RUNNING_DEVICE = 0x0010
Device is running and not suspended.
CUDBG_ERROR_INVALID_ADDRESS = 0x0012
Address is out-of-range.
CUDBG_ERROR_INCOMPATIBLE_API = 0x0013
The requested API is not available.
CUDBG_ERROR_INITIALIZATION_FAILURE = 0x0014
The API could not be initialized.
CUDBG_ERROR_INVALID_GRID = 0x0015
The specified grid is not valid.
CUDBG_ERROR_NO_EVENT_AVAILABLE = 0x0016
The event queue is empty and there is no event left to be processed.
CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED = 0x0017
Some devices were excluded because they have a watchdog associated with them.
CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED = 0x0018
All devices were exclude because they have a watchdog associated with them.
CUDBG_ERROR_INVALID_ATTRIBUTE = 0x0019
Specified attribute does not exist or is incorrect.

CUDBG_ERROR_ZERO_CALL_DEPTH = 0x001a
No function calls have been made on the device.
CUDBG_ERROR_INVALID_CALL_LEVEL = 0x001b
Specified call level is invalid.
CUDBG_ERROR_COMMUNICATION_FAILURE = 0x001c
Communication error between the debugger and the application.
CUDBG_ERROR_INVALID_CONTEXT = 0x001d
Specified context cannot be found.
CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM = 0x001e
Requested address was not originally allocated from device memory (most likely visible in system memory).
CUDBG_ERROR_MEMORY_UNMAPPING_FAILED = 0x001f
Requested address is not mapped and can not be unmapped.
CUDBG_ERROR_INCOMPATIBLE_DISPLAY_DRIVER = 0x0020
The display driver is incompatible with the API.
CUDBG_ERROR_INVALID_MODULE = 0x0021
The specified module is not valid.
CUDBG_ERROR_LANE_NOT_IN_SYSCALL = 0x0022
The specified lane is not inside a device syscall.
CUDBG_ERROR_MEMCHECK_NOT_ENABLED = 0x0023
Memcheck has not been enabled.
CUDBG_ERROR_INVALID_ENVVAR_ARGS = 0x0024
Some environment variable's value is invalid.
CUDBG_ERROR_OS_RESOURCES = 0x0025
Error while allocating resources from the OS.
CUDBG_ERROR_FORK_FAILED = 0x0026
Error while forking the debugger process.
CUDBG_ERROR_NO_DEVICE_AVAILABLE = 0x0027
No CUDA capable device was found.
CUDBG_ERROR_ATTACH_NOT_POSSIBLE = 0x0028
Attaching to the CUDA program is not possible.

2.2. Initialization

CUDBGResult (*CUDBGAPI_st::finalize) ()

Finalize the API and free all memory.

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_COMMUNICATION_FAILURE, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

See also:

initialize

CUDBGResult (*CUDBGAPI_st::initialize) ()

Initialize the API.

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

See also:

finalize

2.3. Device Execution Control

CUDBGResult (*CUDBGAPI_st::resumeDevice) (uint32_t dev)

Resume a suspended CUDA device.

Parameters

dev
- device index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

suspendDevice

singleStepWarp

CUDBGResult (*CUDBGAPI_st::singleStepWarp) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *warpMask)

Single step an individual warp on a suspended CUDA device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

warpMask
- the warps that have been single-stepped

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 4.1.

See also:

resumeDevice

suspendDevice

CUDBGResult (*CUDBGAPI_st::singleStepWarp40) (uint32_t dev, uint32_t sm, uint32_t wp)

Single step an individual warp on a suspended CUDA device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

Deprecated in CUDA 4.1.

See also:

resumeDevice

suspendDevice

singleStepWarp

CUDBGResult (*CUDBGAPI_st::suspendDevice) (uint32_t dev)

Suspends a running CUDA device.

Parameters

dev
- device index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

resumeDevice

singleStepWarp

2.4. Breakpoints

CUDBGResult (*CUDBGAPI_st::setBreakpoint) (uint32_t dev, uint64_t addr)

Sets a breakpoint at the given instruction address for the given device.

Parameters

dev
- the device index

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INVALID_DEVICE

Description

Since CUDA 3.2.

See also:

unsetBreakpoint

CUDBGResult (*CUDBGAPI_st::setBreakpoint31) (uint64_t addr)

Sets a breakpoint at the given instruction address.

Parameters

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

unsetBreakpoint31

CUDBGResult (*CUDBGAPI_st::unsetBreakpoint) (uint32_t dev, uint64_t addr)

Unsets a breakpoint at the given instruction address for the given device.

Parameters

dev
- the device index

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INVALID_DEVICE

Description

Since CUDA 3.2.

See also:

setBreakpoint

CUDBGResult (*CUDBGAPI_st::unsetBreakpoint31) (uint64_t addr)

Unsets a breakpoint at the given instruction address.

Parameters

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

setBreakpoint31

2.5. Device State Inspection

CUDBGResult (*CUDBGAPI_st::memcheckReadErrorAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *address, ptxStorageKind *storage)

Get the address that memcheck detected an error on.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

address
- returned address detected by memcheck

storage
- returned address class of address

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMCHECK_NOT_ENABLED, CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*CUDBGAPI_st::readActiveLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *activeLanesMask)

Reads the bitmask of active lanes on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

activeLanesMask
- the returned bitmask of active lanes

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

CUDBGResult (*CUDBGAPI_st::readBlockIdx) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockIdx)

Reads the CUDA block index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

blockIdx
- the returned CUDA block index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

readGridId

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readBlockIdx32) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *blockIdx)

Reads the two-dimensional CUDA block index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

blockIdx
- the returned CUDA block index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 4.0.

See also:

readGridId

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readBrokenWarps) (uint32_t dev, uint32_t sm, uint64_t *brokenWarpsMask)

Reads the bitmask of warps that are at a breakpoint on a given SM.

Parameters

dev
- device index

sm
- SM index

brokenWarpsMask
- the returned bitmask of broken warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readCallDepth) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)

Reads the call depth (number of calls) for a given lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

depth
- the returned call depth

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

readReturnAddress

readVirtualReturnAddress

CUDBGResult (*CUDBGAPI_st::readCallDepth32) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *depth)

Reads the call depth (number of calls) for a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

depth
- the returned call depth

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readReturnAddress32

readVirtualReturnAddress32

CUDBGResult (*CUDBGAPI_st::readCodeMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the code memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readConstMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the constant memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readGlobalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the global memory segment (entire 40-bit VA on Fermi+).

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Description

Since CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readGlobalMemory31) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the global memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readGridId) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *gridId64)

Reads the 64-bit CUDA grid index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridId64

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 5.5.

See also:

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readGridId50) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)

Reads the CUDA grid index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridId
- the returned CUDA grid index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 5.5.

See also:

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readLaneException) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, CUDBGException_t *exception)

Reads the exception type for a given lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

exception
- the returned exception type

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::readLaneStatus) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, bool *error)

Reads the status of the given lane. For specific error values, use readLaneException.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

error
- true if there is an error

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

CUDBGResult (*CUDBGAPI_st::readLocalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the local memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readParamMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the param memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)

Reads the PC on the given active lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

pc
- the returned PC

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readVirtualPC

CUDBGResult (*CUDBGAPI_st::readPinnedMemory) (uint64_t addr, void *buf, uint32_t sz)

Reads content at pinned address in system memory.

Parameters

addr
- system memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t regno, uint32_t *val)

Reads content of a hardware register.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

regno
- register index

val
- buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readPC

CUDBGResult (*CUDBGAPI_st::readReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)

Reads the physical return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

level
- the specified call level

ra
- the returned return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

readCallDepth

readVirtualReturnAddress

CUDBGResult (*CUDBGAPI_st::readReturnAddress32) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)

Reads the physical return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

level
- the specified call level

ra
- the returned return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readCallDepth32

readVirtualReturnAddress32

CUDBGResult (*CUDBGAPI_st::readSharedMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the shared memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readLocalMemory

readTextureMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readSyscallCallDepth) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)

Reads the call depth of syscalls for a given lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

depth
- the returned call depth

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.1.

See also:

readReturnAddress

readVirtualReturnAddress

CUDBGResult (*CUDBGAPI_st::readTextureMemory) (uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t id, uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)

Read the content of texture memory with given id and coords on sm_20 and lower.

Parameters

devId
- device index

vsm
- SM index

wp
- warp index

id
- texture id (the value of DW_AT_location attribute in the relocated ELF image)

dim
- texture dimension (1 to 4)

coords
- array of coordinates of size dim

buf
- result buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Read the content of texture memory with given id and coords on sm_20 and lower.

On sm_30 and higher, use CUDBGAPI_st::readTextureMemoryBindless instead.

Since CUDA 4.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readTextureMemoryBindless) (uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t texSymtabIndex, uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)

Read the content of texture memory with given symtab index and coords on sm_30 and higher.

Parameters

devId
- device index

vsm
- SM index

wp
- warp index

texSymtabIndex
- global symbol table index of the texture symbol

dim
- texture dimension (1 to 4)

coords
- array of coordinates of size dim

buf
- result buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Read the content of texture memory with given symtab index and coords on sm_30 and higher.

For sm_20 and lower, use CUDBGAPI_st::readTextureMemory instead.

Since CUDA 4.2.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*CUDBGAPI_st::readThreadIdx) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, CuDim3 *threadIdx)

Reads the CUDA thread index running on valid lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

threadIdx
- the returned CUDA thread index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE,

CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readValidLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *validLanesMask)

Reads the bitmask of valid lanes on a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

validLanesMask
- the returned bitmask of valid lanes

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readValidWarps) (uint32_t dev, uint32_t sm, uint64_t *validWarpsMask)

Reads the bitmask of valid warps on a given SM.

Parameters

dev
- device index

sm
- SM index

validWarpsMask
- the returned bitmask of valid warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readBrokenWarps

readValidLanes

readActiveLanes

CUDBGResult (*CUDBGAPI_st::readVirtualPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)

Reads the virtual PC on the given active lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

pc
- the returned PC

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN_FUNCTION

Description

Since CUDA 3.0.

See also:

readPC

CUDBGResult (*CUDBGAPI_st::readVirtualReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)

Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

level
- the specified call level

ra
- the returned virtual return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INTERNAL

Description

Since CUDA 4.0.

See also:

readCallDepth

readReturnAddress

CUDBGResult (*CUDBGAPI_st::readVirtualReturnAddress32) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)

Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

level
- the specified call level

ra
- the returned virtual return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INTERNAL

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readCallDepth32

readReturnAddress32

CUDBGResult (*CUDBGAPI_st::writePinnedMemory) (uint64_t addr, const void *buf, uint32_t sz)

Writes content to pinned address in system memory.

Parameters

addr
- system memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readLocalMemory

readRegister

readPC

2.6. Device State Alteration

CUDBGResult (*CUDBGAPI_st::writeGlobalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the global memory segment (entire 40-bit VA on Fermi+).

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Description

Since CUDA 3.2.

See also:

writeParamMemory

writeSharedMemory

writeLocalMemory

writeRegister

CUDBGResult (*CUDBGAPI_st::writeGlobalMemory31)(uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the global memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

writeParamMemory

writeSharedMemory

writeLocalMemory

writeRegister

CUDBGResult (*CUDBGAPI_st::writeLocalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the local memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeParamMemory

writeSharedMemory

writeRegister

CUDBGResult (*CUDBGAPI_st::writeParamMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the param memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeSharedMemory

writeLocalMemory

writeRegister

CUDBGResult (*CUDBGAPI_st::writeRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t regno, uint32_t val)

Writes content to a hardware register.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

regno
- register index

val
- buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeParamMemory

writeSharedMemory

writeLocalMemory

CUDBGResult (*CUDBGAPI_st::writeSharedMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the shared memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeParamMemory

writeLocalMemory

writeRegister

2.7. Grid Properties

struct CUDBGGridInfo

Grid info.

enum CUDBGGridStatus

Grid status.

Values

CUDBG_GRID_STATUS_INVALID
An invalid grid ID was passed, or an error occurred during status lookup.
CUDBG_GRID_STATUS_PENDING
The grid was launched but is not running on the HW yet.
CUDBG_GRID_STATUS_ACTIVE
The grid is currently running on the HW.
CUDBG_GRID_STATUS_SLEEPING
The grid is on the device, doing a join.
CUDBG_GRID_STATUS_TERMINATED
The grid has finished executing.
CUDBG_GRID_STATUS_UNDETERMINED
The grid is either QUEUED or TERMINATED.

CUDBGResult (*CUDBGAPI_st::getBlockDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockDim)

Get the number of threads in the given block.

Parameters

dev
- device index
sm
- SM index
wp
- warp index
blockDim
- the returned number of threads in the block

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getGridDim

CUDBGResult (*CUDBGAPI_st::getElfImage) (uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint64_t *size)

Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

relocated
- set to true to specify the relocated ELF image, false otherwise

***elfImage**
- pointer to the ELF image

size
- size of the ELF image (64 bits)

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

CUDBGResult (*CUDBGAPI_st::getElfImage32) (uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint32_t *size)

Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

relocated
- set to true to specify the relocated ELF image, false otherwise

***elfImage**
- pointer to the ELF image

size
- size of the ELF image (32 bits)

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 4.0.

CUDBGResult (*CUDBGAPI_st::getGridAttribute) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttribute attr, uint64_t *value)

Get the value of a grid attribute.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

attr
- the attribute

value
- the returned value of the attribute

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::getGridAttributes) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttributeValuePair *pairs, uint32_t numPairs)

Get several grid attribute values in a single API call.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

pairs
- array of attribute/value pairs

numPairs
- the number of attribute/values pairs in the array

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::getGridDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *gridDim)

Get the number of blocks in the given grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridDim
- the returned number of blocks in the grid

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

getBlockDim

CUDBGResult (*CUDBGAPI_st::getGridDim32) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)

Get the number of blocks in the given grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridDim
- the returned number of blocks in the grid

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 4.0.

See also:

getBlockDim

CUDBGResult (*CUDBGAPI_st::getGridInfo) (uint32_t dev, uint64_t gridId64, CUDBGGridInfo *gridInfo)

Get information about the specified grid. If the context of the grid has already been destroyed, the function will return CUDBG_ERROR_INVALID_GRID, although the grid id is correct.

Parameters

dev
gridId64
gridInfo
- pointer to a client allocated structure in which grid info will be returned.

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_SUCCESS

Description

Since CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getGridStatus) (uint32_t dev, uint64_t gridId64, CUDBGGridStatus *status)

Check whether the grid corresponding to the given gridId is still present on the device.

Parameters

dev

gridId64

- 64-bit grid ID

status

- enum indicating whether the grid status is INVALID, PENDING, ACTIVE, SLEEPING, TERMINATED or UNDETERMINED

Returns

CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INTERNAL

Description

Since CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getGridStatus50) (uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)

Check whether the grid corresponding to the given gridId is still present on the device.

Parameters

dev

gridId

- grid ID

status

- enum indicating whether the grid status is INVALID, PENDING, ACTIVE, SLEEPING, TERMINATED or UNDETERMINED

Returns

CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INTERNAL

Description

Since CUDA 5.0.

Deprecated in CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getTID) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *tid)

Get the ID of the Linux thread hosting the context of the grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

tid
- the returned thread id

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

2.8. Device Properties

CUDBGResult (*CUDBGAPI_st::getDeviceType) (uint32_t dev, char *buf, uint32_t sz)

Get the string description of the device.

Parameters

dev
- device index

buf
- the destination buffer

sz
- the size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getSMType

CUDBGResult (*CUDBGAPI_st::getNumDevices) (uint32_t *numDev)

Get the number of installed CUDA devices.

Parameters

numDev
- the returned number of devices

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumSMs

getNumWarps

getNumLanes

getNumRegisters

CUDBGResult (*CUDBGAPI_st::getNumLanes) (uint32_t dev, uint32_t *numLanes)

Get the number of lanes per warp on the device.

Parameters

dev
- device index

numLanes
- the returned number of lanes

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumSMs

getNumWarps

getNumRegisters

CUDBGResult (*CUDBGAPI_st::getNumRegisters) (uint32_t dev, uint32_t *numRegs)

Get the number of registers per lane on the device.

Parameters

dev
- device index

numRegs
- the returned number of registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumSMs

getNumWarps

getNumLanes

CUDBGResult (*CUDBGAPI_st::getNumSMs) (uint32_t dev, uint32_t *numSMs)

Get the total number of SMs on the device.

Parameters

dev
- device index

numSMs
- the returned number of SMs

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumWarps

getNumLanes

getNumRegisters

CUDBGResult (*CUDBGAPI_st::getNumWarps) (uint32_t dev, uint32_t *numWarps)

Get the number of warps per SM on the device.

Parameters

dev
- device index

numWarps
- the returned number of warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumSMs

getNumLanes

getNumRegisters

CUDBGResult (*CUDBGAPI_st::getSmType) (uint32_t dev, char *buf, uint32_t sz)

Get the SM type of the device.

Parameters

dev
- device index

buf
- the destination buffer

sz
- the size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getDeviceType

2.9. DWARF Utilities

CUDBGResult (*CUDBGAPI_st::disassemble) (uint32_t dev, uint64_t addr, uint32_t *instSize, char *buf, uint32_t sz)

Disassemble instruction at instruction address.

Parameters

dev
- device index

addr
- instruction address

instSize
- instruction size (32 or 64 bits)

buf
- disassembled instruction buffer

sz
- disassembled instruction buffer size

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

CUDBGResult (*CUDBGAPI_st::getHostAddrFromDeviceAddr) (uint32_t dev, uint64_t device_addr, uint64_t *host_addr)

given a device virtual address, return a corresponding system memory virtual address.

Parameters

dev
- device index

device_addr
- device memory address

host_addr
- returned system memory address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_CONTEXT, CUDBG_ERROR_INVALID_MEMORY_SEGMENT

Description

Since CUDA 4.1.

See also:

readGlobalMemory

writeGlobalMemory

CUDBGResult (*CUDBGAPI_st::getPhysicalRegister30) (uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)

Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC.

Parameters

pc
- Program counter

reg
- virtual register index

buf
- physical register name(s)

sz
- the physical register name buffer size

numPhysRegs
- number of physical register names returned

regClass
- the class of the physical registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

Deprecated in CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::getPhysicalRegister40) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)

Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC.

Parameters

dev
- device index

sm
- SM index

wp
- warp indx

pc
- Program counter

reg
- virtual register index

buf
- physical register name(s)

sz
- the physical register name buffer size

numPhysRegs
- number of physical register names returned

regClass
- the class of the physical registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

Description

Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC. If a virtual register name is mapped to more than one physical register, the physical register with the lowest physical register index will contain the highest bits of the virtual register, and the physical register with the highest physical register index will contain the lowest bits.

Since CUDA 3.1.

Deprecated in CUDA 4.1.

CUDBGResult (*CUDBGAPI_st::isDeviceCodeAddress) (uintptr_t addr, bool *isDeviceAddress)

Determines whether a virtual address resides within device code.

Parameters

addr
- virtual address

isDeviceAddress
- true if address resides within device code

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Description

Since CUDA 3.0.

CUDBGResult (*CUDBGAPI_st::lookupDeviceCodeSymbol) (char *symName, bool *symFound, uintptr_t *symAddr)

Determines whether a symbol represents a function in device code and returns its virtual address.

Parameters

symName
- symbol name

symFound
- set to true if the symbol is found

symAddr
- the symbol virtual address if found

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Description

Since CUDA 3.0.

2.10. Events

One of those events will create a CUDBGEvent:

- the elf image of the current kernel has been loaded and the addresses within its DWARF sections have been relocated (and can now be used to set breakpoints),
- a device breakpoint has been hit,
- a CUDA kernel is ready to be launched,
- a CUDA kernel has terminated.

When a CUDBGEvent is created, the debugger is notified by calling the callback functions registered with setNotifyNewEventCallback() after the API struct initialization. It is up to the debugger to decide what method is best to be notified. The debugger API routines cannot be called from within the callback function or the routine will return an error.

Upon notification the debugger is responsible for handling the CUDBGEvents in the event queue by using CUDBGAPI_st::getNextEvent(), and for acknowledging the debugger API that the event has been handled by calling CUDBGAPI_st::acknowledgeEvent(). In the case of an event raised by the device itself,

such as a breakpoint being hit, the event queue will be empty. It is the responsibility of the debugger to inspect the hardware any time a CUDBGEvent is received.

Example:

```
CUDBGEvent event;
      CUDBGResult res;
      for (res = cudbgAPI->getNextEvent(&event);
           res == CUDBG_SUCCESS && event.kind != CUDBG_EVENT_INVALID;
           res = cudbgAPI->getNextEvent(&event)) {
          switch (event.kind)
              {
              case CUDBG_EVENT_ELF_IMAGE_LOADED:
                  //...
                  break;
              case CUDBG_EVENT_KERNEL_READY:
                  //...
                  break;
              case CUDBG_EVENT_KERNEL_FINISHED:
                  //...
                  break;
              default:
                  error(...);
              }
          }
```

See cuda-tdep.c and cuda-linux-nat.c files in the cuda-gdb source code for a more detailed example on how to use CUDBGEvent.

struct CUDBGEvent

Event information container.

struct CUDBGEventCallbackData

Event information passed to callback set with setNotifyNewEventCallback function.

struct CUDBGEventCallbackData40

Event information passed to callback set with setNotifyNewEventCallback function.

enum CUDBGEventKind

CUDA Kernel Events.

Values

CUDBG_EVENT_INVALID = 0x000
Invalid event.

CUDBG_EVENT_ELF_IMAGE_LOADED = 0x001
The ELF image for a CUDA source module is available.

CUDBG_EVENT_KERNEL_READY = 0x002
A CUDA kernel is about to be launched.

CUDBG_EVENT_KERNEL_FINISHED = 0x003
A CUDA kernel has terminated.

CUDBG_EVENT_INTERNAL_ERROR = 0x004
An internal error occur. The debugging framework may be unstable.
CUDBG_EVENT_CTX_PUSH = 0x005
A CUDA context was pushed.
CUDBG_EVENT_CTX_POP = 0x006
A CUDA CTX was popped.
CUDBG_EVENT_CTX_CREATE = 0x007
A CUDA CTX was created.
CUDBG_EVENT_CTX_DESTROY = 0x008
A CUDA context was destroyed.
CUDBG_EVENT_TIMEOUT = 0x009
An timeout event is sent at regular interval. This event can safely ge ignored.
CUDBG_EVENT_ATTACH_COMPLETE = 0x00a
The attach process has completed and debugging of device code may start.
CUDBG_EVENT_DETACH_COMPLETE = 0x00b

typedef (*CUDBGNotifyNewEventCallback) (CUDBGEventCallbackData* data)

function type of the function called to notify debugger of the presence of a new event in the event queue.

typedef (*CUDBGNotifyNewEventCallback31) (void* data)

function type of the function called to notify debugger of the presence of a new event in the event queue.

Deprecated in CUDA 3.2.

CUDBGResult (*CUDBGAPI_st::acknowledgeEvent30) (CUDBGEvent30 *event)

Inform the debugger API that the event has been processed.

Parameters

event
- pointer to the event that has been processed

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::acknowledgeEvents42) ()

Inform the debugger API that synchronous events have been processed.

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.1.

Deprecated in CUDA 5.0.

CUDBGResult (*CUDBGAPI_st::acknowledgeSyncEvents) ()

Inform the debugger API that synchronous events have been processed.

Returns

CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*CUDBGAPI_st::getNextAsyncEvent) (CUDBGEvent *event)

Copies the next available event in the asynchronous event queue into 'event' and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getNextAsyncEvent50) (CUDBGEvent50 *event)

Copies the next available event in the asynchronous event queue into 'event' and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.0.

Deprecated in CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getNextEvent30) (CUDBGEvent30 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::getNextEvent32) (CUDBGEvent32 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0

CUDBGResult (*CUDBGAPI_st::getNextEvent42) (CUDBGEvent42 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 4.0.

Deprecated in CUDA 5.0

CUDBGResult (*CUDBGAPI_st::getNextSyncEvent) (CUDBGEvent *event)

Copies the next available event in the synchronous event queue into 'event' and removes it from the queue.

Parameters

event
- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getNextSyncEvent50) (CUDBGEvent50 *event)

Parameters

event
- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.0.

Deprecated in CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::setNotifyNewEventCallback) (CUDBGNotifyNewEventCallback callback)

Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters

callback
- the callback function

Returns

CUDBG_SUCCESS

Description

Since CUDA 4.1.

CUDBGResult (*CUDBGAPI_st::setNotifyNewEventCallback31) (CUDBGNotifyNewEventCallback31 callback, void *data)

Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters

callback
- the callback function

data
- a pointer to be passed to the callback when called

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

CUDBGResult (*CUDBGAPI_st::setNotifyNewEventCallback40) (CUDBGNotifyNewEventCallback40 callback)

Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters

callback
- the callback function

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.2.

Deprecated in CUDA 4.1.

Chapter 3. DATA STRUCTURES

Here are the data structures with brief descriptions:

cudbgGetAPI
The CUDA debugger API routines

CUDBGEvent
Event information container

CUDBGEvent::CUDBGEvent::cases_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st

CUDBGEventCallbackData
Event information passed to callback set with setNotifyNewEventCallback function

CUDBGEventCallbackData40
Event information passed to callback set with setNotifyNewEventCallback function

CUDBGGridInfo
Grid info

3.1. CUDBGAPI_st Struct Reference

The CUDA debugger API routines.

CUDBGResult (*acknowledgeEvent30) (CUDBGEvent30 *event)

Inform the debugger API that the event has been processed.

Parameters

event

- pointer to the event that has been processed

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.1.

CUDBGResult (*acknowledgeEvents42) ()

Inform the debugger API that synchronous events have been processed.

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.1.

Deprecated in CUDA 5.0.

CUDBGResult (*acknowledgeSyncEvents) ()

Inform the debugger API that synchronous events have been processed.

Returns

CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*clearAttachState) ()

Clear attach-specific state prior to detach.

Returns

CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*disassemble) (uint32_t dev, uint64_t addr, uint32_t *instSize, char *buf, uint32_t sz)

Disassemble instruction at instruction address.

Parameters

dev
- device index

addr
- instruction address

instSize
- instruction size (32 or 64 bits)

buf
- disassembled instruction buffer

sz
- disassembled instruction buffer size

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

CUDBGResult (*finalize) ()

Finalize the API and free all memory.

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_COMMUNICATION_FAILURE, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

See also:

initialize

CUDBGResult (*getBlockDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockDim)

Get the number of threads in the given block.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

blockDim
- the returned number of threads in the block

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getGridDim

CUDBGResult (*getDevicePCIBusInfo) (uint32_t devId, uint32_t *pciBusId, uint32_t *pciDevId)

Get PCI bus and device ids associated with device devId.

Parameters

devId
- the cuda device id

pciBusId
- pointer where corresponding PCI BUS ID would be stored

pciDevId
- pointer where corresponding PCI DEVICE ID would be stored

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE

CUDBGResult (*getDeviceType) (uint32_t dev, char *buf, uint32_t sz)

Get the string description of the device.

Parameters

dev
- device index

buf
- the destination buffer

sz
- the size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getSMType

CUDBGResult (*getElfImage) (uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint64_t *size)

Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

relocated
- set to true to specify the relocated ELF image, false otherwise

***elfImage**
- pointer to the ELF image

size
- size of the ELF image (64 bits)

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

CUDBGResult (*getElfImage32) (uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint32_t *size)

Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

relocated
- set to true to specify the relocated ELF image, false otherwise

***elfImage**
- pointer to the ELF image

size
- size of the ELF image (32 bits)

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 4.0.

CUDBGResult (*getGridAttribute) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttribute attr, uint64_t *value)

Get the value of a grid attribute.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

attr
- the attribute

value
- the returned value of the attribute

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

CUDBGResult (*getGridAttributes) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttributeValuePair *pairs, uint32_t numPairs)

Get several grid attribute values in a single API call.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

pairs
- array of attribute/value pairs

numPairs
- the number of attribute/values pairs in the array

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

CUDBGResult (*getGridDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *gridDim)

Get the number of blocks in the given grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridDim
- the returned number of blocks in the grid

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

getBlockDim

CUDBGResult (*getGridDim32) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)

Get the number of blocks in the given grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridDim
- the returned number of blocks in the grid

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 4.0.

See also:

getBlockDim

CUDBGResult (*getGridInfo) (uint32_t dev, uint64_t gridId64, CUDBGGridInfo *gridInfo)

Get information about the specified grid. If the context of the grid has already been destroyed, the function will return CUDBG_ERROR_INVALID_GRID, although the grid id is correct.

Parameters

dev
gridId64
gridInfo
- pointer to a client allocated structure in which grid info will be returned.

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_SUCCESS

Description

Since CUDA 5.5.

CUDBGResult (*getGridStatus) (uint32_t dev, uint64_t gridId64, CUDBGGridStatus *status)

Check whether the grid corresponding to the given gridId is still present on the device.

Parameters

dev
gridId64
- 64-bit grid ID

status
- enum indicating whether the grid status is INVALID, PENDING, ACTIVE, SLEEPING, TERMINATED or UNDETERMINED

Returns

CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INTERNAL

Description

Since CUDA 5.5.

CUDBGResult (*getGridStatus50) (uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)

Check whether the grid corresponding to the given gridId is still present on the device.

Parameters

dev

gridId

- grid ID

status

- enum indicating whether the grid status is INVALID, PENDING, ACTIVE, SLEEPING, TERMINATED or UNDETERMINED

Returns

CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INTERNAL

Description

Since CUDA 5.0.

Deprecated in CUDA 5.5.

CUDBGResult (*getHostAddrFromDeviceAddr) (uint32_t dev, uint64_t device_addr, uint64_t *host_addr)

given a device virtual address, return a corresponding system memory virtual address.

Parameters

dev

- device index

device_addr

- device memory address

host_addr

- returned system memory address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_CONTEXT, CUDBG_ERROR_INVALID_MEMORY_SEGMENT

Description

Since CUDA 4.1.

See also:

readGlobalMemory

writeGlobalMemory

CUDBGResult (*getNextAsyncEvent) (CUDBGEvent *event)

Copies the next available event in the asynchronous event queue into 'event' and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

Parameters

event
- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.5.

CUDBGResult (*getNextAsyncEvent50) (CUDBGEvent50 *event)

Copies the next available event in the asynchronous event queue into 'event' and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

Parameters

event
- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.0.

Deprecated in CUDA 5.5.

CUDBGResult (*getNextEvent30) (CUDBGEvent30 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

Parameters

event
- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.1.

CUDBGResult (*getNextEvent32) (CUDBGEvent32 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

Parameters

event
- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0

CUDBGResult (*getNextEvent42) (CUDBGEvent42 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 4.0.

Deprecated in CUDA 5.0

CUDBGResult (*getNextSyncEvent) (CUDBGEvent *event)

Copies the next available event in the synchronous event queue into 'event' and removes it from the queue.

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.5.

CUDBGResult (*getNextSyncEvent50) (CUDBGEvent50 *event)

Parameters

event

- pointer to an event container where to copy the event parameters

Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Description

Since CUDA 5.0.

Deprecated in CUDA 5.5.

CUDBGResult (*getNumDevices) (uint32_t *numDev)

Get the number of installed CUDA devices.

Parameters

numDev
- the returned number of devices

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumSMs

getNumWarps

getNumLanes

getNumRegisters

CUDBGResult (*getNumLanes) (uint32_t dev, uint32_t *numLanes)

Get the number of lanes per warp on the device.

Parameters

dev
- device index

numLanes
- the returned number of lanes

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumSMs

getNumWarps

getNumRegisters

CUDBGResult (*getNumRegisters) (uint32_t dev, uint32_t *numRegs)

Get the number of registers per lane on the device.

Parameters

dev
- device index

numRegs
- the returned number of registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumSMs

getNumWarps

getNumLanes

CUDBGResult (*getNumSMs) (uint32_t dev, uint32_t *numSMs)

Get the total number of SMs on the device.

Parameters

dev
- device index

numSMs
- the returned number of SMs

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumWarps

getNumLanes

getNumRegisters

CUDBGResult (*getNumWarps) (uint32_t dev, uint32_t *numWarps)

Get the number of warps per SM on the device.

Parameters

dev
- device index

numWarps
- the returned number of warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getNumDevices

getNumSMs

getNumLanes

getNumRegisters

CUDBGResult (*getPhysicalRegister30) (uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)

Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC.

Parameters

pc
- Program counter

reg
- virtual register index

buf
- physical register name(s)

sz
- the physical register name buffer size

numPhysRegs
- number of physical register names returned

regClass
- the class of the physical registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

Deprecated in CUDA 3.1.

CUDBGResult (*getPhysicalRegister40) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)

Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC.

Parameters

dev
- device index

sm
- SM index

wp
- warp indx

pc
- Program counter

reg
- virtual register index

buf
- physical register name(s)

sz
- the physical register name buffer size

numPhysRegs
- number of physical register names returned

regClass
- the class of the physical registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

Description

Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC. If a virtual register name is mapped to more than one physical register, the physical register with the lowest physical register index will

contain the highest bits of the virtual register, and the physical register with the highest physical register index will contain the lowest bits.

Since CUDA 3.1.

Deprecated in CUDA 4.1.

CUDBGResult (*getSmType) (uint32_t dev, char *buf, uint32_t sz)

Get the SM type of the device.

Parameters

dev
- device index

buf
- the destination buffer

sz
- the size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

getDeviceType

CUDBGResult (*getTID) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *tid)

Get the ID of the Linux thread hosting the context of the grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

tid
- the returned thread id

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

CUDBGResult (*initialize) ()

Initialize the API.

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

See also:

finalize

CUDBGResult (*initializeAttachStub) ()

Initialize the attach stub.

Returns

CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*isDeviceCodeAddress) (uintptr_t addr, bool *isDeviceAddress)

Determines whether a virtual address resides within device code.

Parameters

addr
- virtual address

isDeviceAddress
- true if address resides within device code

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Description

Since CUDA 3.0.

CUDBGResult (*lookupDeviceCodeSymbol) (char *symName, bool *symFound, uintptr_t *symAddr)

Determines whether a symbol represents a function in device code and returns its virtual address.

Parameters

symName
- symbol name

symFound
- set to true if the symbol is found

symAddr
- the symbol virtual address if found

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Description

Since CUDA 3.0.

CUDBGResult (*memcheckReadErrorAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *address, ptxStorageKind *storage)

Get the address that memcheck detected an error on.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

address
- returned address detected by memcheck

storage
- returned address class of address

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMCHECK_NOT_ENABLED, CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*readActiveLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *activeLanesMask)

Reads the bitmask of active lanes on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

activeLanesMask
- the returned bitmask of active lanes

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

CUDBGResult (*readBlockIdx) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockIdx)

Reads the CUDA block index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

blockIdx
- the returned CUDA block index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

readGridId

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*readBlockIdx32) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *blockIdx)

Reads the two-dimensional CUDA block index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

blockIdx
- the returned CUDA block index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 4.0.

See also:

readGridId

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*readBrokenWarps) (uint32_t dev, uint32_t sm, uint64_t *brokenWarpsMask)

Reads the bitmask of warps that are at a breakpoint on a given SM.

Parameters

dev
- device index

sm
- SM index

brokenWarpsMask
- the returned bitmask of broken warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*readCallDepth) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)

Reads the call depth (number of calls) for a given lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

depth
- the returned call depth

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

readReturnAddress

readVirtualReturnAddress

CUDBGResult (*readCallDepth32) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *depth)

Reads the call depth (number of calls) for a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

depth
- the returned call depth

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readReturnAddress32

readVirtualReturnAddress32

CUDBGResult (*readCodeMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the code memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readConstMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the constant memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readDeviceExceptionState) (uint32_t devId, uint64_t *exceptionSMMask)

Get the exception state of the SMs on the device.

Parameters

devId
- the cuda device id

exceptionSMMask
- Bit field containing a 1 at (1 << i) if SM i hit an exception

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE

CUDBGResult (*readGlobalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the global memory segment (entire 40-bit VA on Fermi+).

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Description

Since CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readGlobalMemory31) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the global memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readGridId) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *gridId64)

Reads the 64-bit CUDA grid index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridId64

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 5.5.

See also:

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*readGridId50) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)

Reads the CUDA grid index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridId
- the returned CUDA grid index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 5.5.

See also:

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*readLaneException) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, CUDBGException_t *exception)

Reads the exception type for a given lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

exception
- the returned exception type

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

CUDBGResult (*readLaneStatus) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, bool *error)

Reads the status of the given lane. For specific error values, use readLaneException.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

error
- true if there is an error

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

CUDBGResult (*readLocalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the local memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readRegister

readPC

CUDBGResult (*readParamMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the param memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)

Reads the PC on the given active lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

pc
- the returned PC

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readVirtualPC

CUDBGResult (*readPinnedMemory) (uint64_t addr, void *buf, uint32_t sz)

Reads content at pinned address in system memory.

Parameters

addr
- system memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t regno, uint32_t *val)

Reads content of a hardware register.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

regno
- register index

val
- buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readPC

CUDBGResult (*readReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)

Reads the physical return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

level
- the specified call level

ra
- the returned return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.0.

See also:

readCallDepth

readVirtualReturnAddress

CUDBGResult (*readReturnAddress32) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)

Reads the physical return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

level
- the specified call level

ra
- the returned return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readCallDepth32

readVirtualReturnAddress32

CUDBGResult (*readSharedMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the shared memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readLocalMemory

readTextureMemory

readRegister

readPC

CUDBGResult (*readSyscallCallDepth) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)

Reads the call depth of syscalls for a given lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

depth
- the returned call depth

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 4.1.

See also:

readReturnAddress

readVirtualReturnAddress

CUDBGResult (*readTextureMemory) (uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t id, uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)

Read the content of texture memory with given id and coords on sm_20 and lower.

Parameters

devId
- device index

vsm
- SM index

wp
- warp index

id
- texture id (the value of DW_AT_location attribute in the relocated ELF image)

dim
- texture dimension (1 to 4)

coords
- array of coordinates of size dim

buf
- result buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Read the content of texture memory with given id and coords on sm_20 and lower.

On sm_30 and higher, use CUDBGAPI_st::readTextureMemoryBindless instead.

Since CUDA 4.0.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readTextureMemoryBindless) (uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t texSymtabIndex, uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)

Read the content of texture memory with given symtab index and coords on sm_30 and higher.

Parameters

devId
- device index

vsm
- SM index

wp
- warp index

texSymtabIndex
- global symbol table index of the texture symbol

dim
- texture dimension (1 to 4)

coords
- array of coordinates of size dim

buf
- result buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Read the content of texture memory with given symtab index and coords on sm_30 and higher.

For sm_20 and lower, use CUDBGAPI_st::readTextureMemory instead.

Since CUDA 4.2.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*readThreadIdx) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, CuDim3 *threadIdx)

Reads the CUDA thread index running on valid lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

threadIdx
- the returned CUDA thread index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE,

CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readBrokenWarps

readValidWarps

readValidLanes

readActiveLanes

CUDBGResult (*readValidLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *validLanesMask)

Reads the bitmask of valid lanes on a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

validLanesMask
- the returned bitmask of valid lanes

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readBrokenWarps

readValidWarps

readActiveLanes

CUDBGResult (*readValidWarps) (uint32_t dev, uint32_t sm, uint64_t *validWarpsMask)

Reads the bitmask of valid warps on a given SM.

Parameters

dev
- device index

sm
- SM index

validWarpsMask
- the returned bitmask of valid warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

readGridId

readBlockIdx

readThreadIdx

readBrokenWarps

readValidLanes

readActiveLanes

CUDBGResult (*readVirtualPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)

Reads the virtual PC on the given active lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

pc
- the returned PC

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN_FUNCTION

Description

Since CUDA 3.0.

See also:

readPC

CUDBGResult (*readVirtualReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)

Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

level
- the specified call level

ra
- the returned virtual return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INTERNAL

Description

Since CUDA 4.0.

See also:

readCallDepth

readReturnAddress

CUDBGResult (*readVirtualReturnAddress32) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)

Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

level
- the specified call level

ra
- the returned virtual return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_CALL_LEVEL, CUDBG_ERROR_ZERO_CALL_DEPTH, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INTERNAL

Description

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readCallDepth32

readReturnAddress32

CUDBGResult (*requestCleanupOnDetach) ()

Request for cleanup of driver state when detaching.

Returns

CUDBG_SUCCESS

Description

Since CUDA 5.0.

CUDBGResult (*resumeDevice) (uint32_t dev)

Resume a suspended CUDA device.

Parameters

dev
- device index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

suspendDevice

singleStepWarp

CUDBGResult (*setBreakpoint) (uint32_t dev, uint64_t addr)

Sets a breakpoint at the given instruction address for the given device.

Parameters

dev
- the device index

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INVALID_DEVICE

Description

Since CUDA 3.2.

See also:

unsetBreakpoint

CUDBGResult (*setBreakpoint31) (uint64_t addr)

Sets a breakpoint at the given instruction address.

Parameters

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

unsetBreakpoint31

CUDBGResult (*setKernelLaunchNotificationMode) (CUDBGKernelLaunchNotifyMode mode)

Set the launch notification policy.

Parameters

mode
- mode to deliver kernel launch notifications in

Returns

CUDBG_SUCCESS

Description

Since CUDA 5.5.

CUDBGResult (*setNotifyNewEventCallback) (CUDBGNotifyNewEventCallback callback)

Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters

callback
- the callback function

Returns

CUDBG_SUCCESS

Description

Since CUDA 4.1.

CUDBGResult (*setNotifyNewEventCallback31) (CUDBGNotifyNewEventCallback31 callback, void *data)

Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters

callback
- the callback function

data
- a pointer to be passed to the callback when called

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

CUDBGResult (*setNotifyNewEventCallback40) (CUDBGNotifyNewEventCallback40 callback)

Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters

callback
- the callback function

Returns

CUDBG_SUCCESS

Description

Since CUDA 3.2.

Deprecated in CUDA 4.1.

CUDBGResult (*singleStepWarp) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *warpMask)

Single step an individual warp on a suspended CUDA device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

warpMask
- the warps that have been single-stepped

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 4.1.

See also:

resumeDevice

suspendDevice

CUDBGResult (*singleStepWarp40) (uint32_t dev, uint32_t sm, uint32_t wp)

Single step an individual warp on a suspended CUDA device.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN

Description

Since CUDA 3.0.

Deprecated in CUDA 4.1.

See also:

resumeDevice

suspendDevice

singleStepWarp

CUDBGResult (*suspendDevice) (uint32_t dev)

Suspends a running CUDA device.

Parameters

dev
- device index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

resumeDevice

singleStepWarp

CUDBGResult (*unsetBreakpoint) (uint32_t dev, uint64_t addr)

Unsets a breakpoint at the given instruction address for the given device.

Parameters

dev
- the device index

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INVALID_DEVICE

Description

Since CUDA 3.2.

See also:

setBreakpoint

CUDBGResult (*unsetBreakpoint31) (uint64_t addr)

Unsets a breakpoint at the given instruction address.

Parameters

addr
- instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

setBreakpoint31

CUDBGResult (*writeGlobalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the global memory segment (entire 40-bit VA on Fermi+).

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Description

Since CUDA 3.2.

See also:

writeParamMemory

writeSharedMemory

writeLocalMemory

writeRegister

CUDBGResult (*writeGlobalMemory31) (uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the global memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

Deprecated in CUDA 3.2.

See also:

writeParamMemory

writeSharedMemory

writeLocalMemory

writeRegister

CUDBGResult (*writeLocalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the local memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeParamMemory

writeSharedMemory

writeRegister

CUDBGResult (*writeParamMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the param memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeSharedMemory

writeLocalMemory

writeRegister

CUDBGResult (*writePinnedMemory) (uint64_t addr, const void *buf, uint32_t sz)

Writes content to pinned address in system memory.

Parameters

addr
- system memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.2.

See also:

readCodeMemory

readConstMemory

readGlobalMemory

readParamMemory

readSharedMemory

readLocalMemory

readRegister

readPC

CUDBGResult (*writeRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t regno, uint32_t val)

Writes content to a hardware register.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

regno
- register index

val
- buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeParamMemory

writeSharedMemory

writeLocalMemory

CUDBGResult (*writeSharedMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the shared memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Description

Since CUDA 3.0.

See also:

writeGlobalMemory

writeParamMemory

writeLocalMemory

writeRegister

3.2. CUDBGEvent Struct Reference

Event information container.

CUDBGEvent::cases

Information for each type of event.

CUDBGEventKind CUDBGEvent::kind

Event type.

3.3. CUDBGEvent::cases_st Union Reference

struct CUDBGEvent::cases_st::contextCreate_st CUDBGEvent::cases_st::contextCreate

Information about the context being created.

struct CUDBGEvent::cases_st::contextDestroy_st CUDBGEvent::cases_st::contextDestroy

Information about the context being destroyed.

struct CUDBGEvent::cases_st::contextPop_st CUDBGEvent::cases_st::contextPop

Information about the context being popped.

struct CUDBGEvent::cases_st::contextPush_st CUDBGEvent::cases_st::contextPush

Information about the context being pushed.

struct CUDBGEvent::cases_st::elfImageLoaded_st CUDBGEvent::cases_st::elfImageLoaded

Information about the loaded ELF image.

struct CUDBGEvent::cases_st::internalError_st CUDBGEvent::cases_st::internalError

Information about internal erros.

struct CUDBGEvent::cases_st::kernelFinished_st CUDBGEvent::cases_st::kernelFinished

Information about the kernel that just terminated.

struct CUDBGEvent::cases_st::kernelReady_st CUDBGEvent::cases_st::kernelReady

Information about the kernel ready to be launched.

3.4. CUDBGEvent::cases_st::contextCreate_st Struct Reference

uint64_t CUDBGEvent::cases_st::contextCreate_st::context

the context being created.

uint32_t CUDBGEvent::cases_st::contextCreate_st::dev

device index of the context.

uint32_t CUDBGEvent::cases_st::contextCreate_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

3.5. CUDBGEvent::cases_st::contextDestroy_st Struct Reference

uint64_t CUDBGEvent::cases_st::contextDestroy_st::context

the context being destroyed.

uint32_t CUDBGEvent::cases_st::contextDestroy_st::dev

device index of the context.

uint32_t CUDBGEvent::cases_st::contextDestroy_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

3.6. CUDBGEvent::cases_st::contextPop_st Struct Reference

uint64_t CUDBGEvent::cases_st::contextPop_st::context

the context being popped.

uint32_t CUDBGEvent::cases_st::contextPop_st::dev

device index of the context.

uint32_t CUDBGEvent::cases_st::contextPop_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

3.7. CUDBGEvent::cases_st::contextPush_st Struct Reference

uint64_t CUDBGEvent::cases_st::contextPush_st::context

the context being pushed.

uint32_t CUDBGEvent::cases_st::contextPush_st::dev

device index of the context.

uint32_t CUDBGEvent::cases_st::contextPush_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

3.8. CUDBGEvent::cases_st::elfImageLoaded_st Struct Reference

uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::context

context of the kernel.

uint32_t CUDBGEvent::cases_st::elfImageLoaded_st::dev

device index of the kernel.

uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::module

module of the kernel.

char *CUDBGEvent::cases_st::elfImageLoaded_st::nonRelocatedElfImag

pointer to the non-relocated ELF image for a CUDA source module.

char *CUDBGEvent::cases_st::elfImageLoaded_st::relocatedElfImage

pointer to the relocated ELF image for a CUDA source module.

uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::size

size of the ELF image (64-bit).

uint32_t CUDBGEvent::cases_st::elfImageLoaded_st::size32

size of the ELF image (32-bit).

Description

Deprecated in CUDA 4.0.

3.9. CUDBGEvent::cases_st::internalError_st Struct Reference

CUDBGResult CUDBGEvent::cases_st::internalError_st::errorType

Type of the internal error.

3.10. CUDBGEvent::cases_st::kernelFinished_st Struct Reference

uint64_t CUDBGEvent::cases_st::kernelFinished_st::context

context of the kernel.

uint32_t CUDBGEvent::cases_st::kernelFinished_st::dev

device index of the kernel.

uint64_t CUDBGEvent::cases_st::kernelFinished_st::function

function of the kernel.

uint64_t CUDBGEvent::cases_st::kernelFinished_st::functionEntry

entry PC of the kernel.

uint32_t CUDBGEvent::cases_st::kernelFinished_st::gridId

grid index of the kernel.

uint64_t CUDBGEvent::cases_st::kernelFinished_st::gridId64

64-bit grid index of the kernel.

uint64_t CUDBGEvent::cases_st::kernelFinished_st::module

module of the kernel.

uint32_t CUDBGEvent::cases_st::kernelFinished_st::tid

host thread id (or LWP id) of the thread hosting the kernel (Linux only).

3.11. CUDBGEvent::cases_st::kernelReady_st Struct Reference

CuDim3 CUDBGEvent::cases_st::kernelReady_st::blockDim

block dimensions of the kernel.

uint64_t CUDBGEvent::cases_st::kernelReady_st::context

context of the kernel.

uint32_t CUDBGEvent::cases_st::kernelReady_st::dev

device index of the kernel.

uint64_t CUDBGEvent::cases_st::kernelReady_st::function

function of the kernel.

uint64_t CUDBGEvent::cases_st::kernelReady_st::functionEntry

entry PC of the kernel.

CuDim3 CUDBGEvent::cases_st::kernelReady_st::gridDim

grid dimensions of the kernel.

uint32_t CUDBGEvent::cases_st::kernelReady_st::gridId

grid index of the kernel.

uint64_t CUDBGEvent::cases_st::kernelReady_st::gridId64

64-bit grid index of the kernel.

uint64_t CUDBGEvent::cases_st::kernelReady_st::module

module of the kernel.

uint64_t CUDBGEvent::cases_st::kernelReady_st::parentGridId

64-bit grid index of the parent grid.

uint32_t CUDBGEvent::cases_st::kernelReady_st::tid

host thread id (or LWP id) of the thread hosting the kernel (Linux only).

CUDBGKernelType CUDBGEvent::cases_st::kernelReady_st::type

the type of the kernel: system or application.

3.12. CUDBGEventCallbackData Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function.

uint32_t CUDBGEventCallbackData::tid

Host thread id of the context generating the event. Zero if not available.

uint32_t CUDBGEventCallbackData::timeout

A boolean notifying the debugger that the debug API timed while waiting for a reponse from the debugger to a previous event. It is up to the debugger to decide what to do in response to a timeout.

3.13. CUDBGEventCallbackData40 Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function.

Deprecated in CUDA 4.1.

uint32_t CUDBGEventCallbackData40::tid

Host thread id of the context generating the event. Zero if not available.

3.14. CUDBGGridInfo Struct Reference

Grid info.

CuDim3 CUDBGGridInfo::blockDim

The block dimensions.

uint64_t CUDBGGridInfo::context

The context this grid belongs to.

uint32_t CUDBGGridInfo::dev

The index of the device this grid is running on.

uint64_t CUDBGGridInfo::function

The function corresponding to this grid.

uint64_t CUDBGGridInfo::functionEntry

The entry address of the function corresponding to this grid.

CuDim3 CUDBGGridInfo::gridDim

The grid dimensions.

uint64_t CUDBGGridInfo::gridId64

The 64-bit grid ID of this grid.

uint64_t CUDBGGridInfo::module

The module this grid belongs to.

CUDBGKernelOrigin CUDBGGridInfo::origin

The origin of this grid, CPU or GPU.

uint64_t CUDBGGridInfo::parentGridId

The 64-bit grid ID that launched this grid.

uint32_t CUDBGGridInfo::tid

The host thread ID that launched this grid.

CUDBGKernelType CUDBGGridInfo::type

The type of the grid.

Chapter 4. DATA FIELDS

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

A

acknowledgeEvent30
cudbgGetAPI

acknowledgeEvents42
cudbgGetAPI

acknowledgeSyncEvents
cudbgGetAPI

B

blockDim
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGGridInfo

C

cases
CUDBGEvent

clearAttachState
cudbgGetAPI

context
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
CUDBGGridInfo
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st

getDevicePCIBusInfo
cudbgGetAPI
getDeviceType
cudbgGetAPI
getElfImage
cudbgGetAPI
getElfImage32
cudbgGetAPI
getGridAttribute
cudbgGetAPI
getGridAttributes
cudbgGetAPI
getGridDim
cudbgGetAPI
getGridDim32
cudbgGetAPI
getGridInfo
cudbgGetAPI
getGridStatus
cudbgGetAPI
getGridStatus50
cudbgGetAPI
getHostAddrFromDeviceAddr
cudbgGetAPI
getNextAsyncEvent
cudbgGetAPI
getNextAsyncEvent50
cudbgGetAPI
getNextEvent30
cudbgGetAPI
getNextEvent32
cudbgGetAPI
getNextEvent42
cudbgGetAPI
getNextSyncEvent
cudbgGetAPI
getNextSyncEvent50
cudbgGetAPI
getNumDevices
cudbgGetAPI
getNumLanes
cudbgGetAPI

readDeviceExceptionState
cudbgGetAPI
readGlobalMemory
cudbgGetAPI
readGlobalMemory31
cudbgGetAPI
readGridId
cudbgGetAPI
readGridId50
cudbgGetAPI
readLaneException
cudbgGetAPI
readLaneStatus
cudbgGetAPI
readLocalMemory
cudbgGetAPI
readParamMemory
cudbgGetAPI
readPC
cudbgGetAPI
readPinnedMemory
cudbgGetAPI
readRegister
cudbgGetAPI
readReturnAddress
cudbgGetAPI
readReturnAddress32
cudbgGetAPI
readSharedMemory
cudbgGetAPI
readSyscallCallDepth
cudbgGetAPI
readTextureMemory
cudbgGetAPI
readTextureMemoryBindless
cudbgGetAPI
readThreadIdx
cudbgGetAPI
readValidLanes
cudbgGetAPI
readValidWarps
cudbgGetAPI

Chapter 5. FILE LIST

Here is a list of all documented files with brief descriptions:

cudadebugger.h
Header file for the CUDA debugger API

5.1. cudadebugger.h

Header file for the CUDA debugger API.

cudadebugger.h

```
/*
 * Copyright 2007-2013 NVIDIA Corporation.  All rights reserved.
 *
 * NOTICE TO LICENSEE:
 *
 * This source code and/or documentation ("Licensed Deliverables") are
 * subject to NVIDIA intellectual property rights under U.S. and
 * international Copyright laws.
 *
 * These Licensed Deliverables contained herein is PROPRIETARY and
 * CONFIDENTIAL to NVIDIA and is being provided under the terms and
 * conditions of a form of NVIDIA software license agreement by and
 * between NVIDIA and Licensee ("License Agreement") or electronically
 * accepted by Licensee.  Notwithstanding any terms or conditions to
 * the contrary in the License Agreement, reproduction or disclosure
 * of the Licensed Deliverables to any third party without the express
 * written consent of NVIDIA is prohibited.
 *
 * NOTWITHSTANDING ANY TERMS OR CONDITIONS TO THE CONTRARY IN THE
 * LICENSE AGREEMENT, NVIDIA MAKES NO REPRESENTATION ABOUT THE
 * SUITABILITY OF THESE LICENSED DELIVERABLES FOR ANY PURPOSE.  IT IS
 * PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND.
 * NVIDIA DISCLAIMS ALL WARRANTIES WITH REGARD TO THESE LICENSED
 * DELIVERABLES, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY,
 * NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.
 * NOTWITHSTANDING ANY TERMS OR CONDITIONS TO THE CONTRARY IN THE
 * LICENSE AGREEMENT, IN NO EVENT SHALL NVIDIA BE LIABLE FOR ANY
 * SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, OR ANY
 * DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
 * WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
 * ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
 * OF THESE LICENSED DELIVERABLES.
```

```
 *
 * U.S. Government End Users.  These Licensed Deliverables are a
 * "commercial item" as that term is defined at 48 C.F.R. 2.101 (OCT
 * 1995), consisting of "commercial computer software" and "commercial
 * computer software documentation" as such terms are used in 48
 * C.F.R. 12.212 (SEPT 1995) and is provided to the U.S. Government
 * only as a commercial end item.  Consistent with 48 C.F.R.12.212 and
 * 48 C.F.R. 227.7202-1 through 227.7202-4 (JUNE 1995), all
 * U.S. Government End Users acquire the Licensed Deliverables with
 * only those rights set forth herein.
 *
 * Any use of the Licensed Deliverables in individual and commercial
 * software must include, in the user documentation and internal
 * comments to the code, the above Disclaimer and U.S. Government End
 * Users Notice.
 */

/*---------------------------------- Includes
----------------------------------*/

#ifndef CUDADEBUGGER_H
#define CUDADEBUGGER_H

#include <stdlib.h>
#include "cuda_stdint.h"

#if defined(__STDC__)
#include <inttypes.h>
#include <stdbool.h>
#endif

#ifdef __cplusplus
extern "C" {
#endif

#if defined(_WIN32) && !defined(_WIN64)
/* Windows 32-bit */
#define PRIxPTR "I32x"
#endif

#if defined(_WIN64)
/* Windows 64-bit */
#define PRIxPTR "I64x"
#endif

#if defined(_WIN32)
/* Windows 32- and 64-bit */
#define PRIx64  "I64x"
#define PRId64  "I64d"
typedef unsigned char bool;
#undef false
#undef true
#define false 0
#define true  1
#endif

/*---------------------------------- API Version
----------------------------------*/

#define CUDBG_API_VERSION_MAJOR      5  /* Major release version number */
#define CUDBG_API_VERSION_MINOR      5  /* Minor release version number */
#define CUDBG_API_VERSION_REVISION  98  /* Revision (build) number */

/*---------------------------------- Constants
----------------------------------*/

#define CUDBG_MAX_DEVICES 32  /* Maximum number of supported devices */
```

```
     #define CUDBG_MAX_SMS     64  /* Maximum number of SMs per device */
     #define CUDBG_MAX_WARPS   64  /* Maximum number of warps per SM */
     #define CUDBG_MAX_LANES   32  /* Maximum number of lanes per warp */

     /*----------------------- Thread/Block Coordinates Types
---------------------*/

     typedef struct { uint32_t x, y; }    CuDim2;   /* DEPRECATED */
     typedef struct { uint32_t x, y, z; } CuDim3;   /* 3-dimensional
coordinates for threads,... */

     /*---------------------- Memory Segments (as used in DWARF)
-------------------*/

     typedef enum {
         ptxUNSPECIFIEDStorage,
         ptxCodeStorage,
         ptxRegStorage,
         ptxSregStorage,
         ptxConstStorage,
         ptxGlobalStorage,
         ptxLocalStorage,
         ptxParamStorage,
         ptxSharedStorage,
         ptxSurfStorage,
         ptxTexStorage,
         ptxTexSamplerStorage,
         ptxGenericStorage,
         ptxIParamStorage,
         ptxOParamStorage,
         ptxFrameStorage,
         ptxMAXStorage
     } ptxStorageKind;

     /*--------------------------- Debugger System Calls
-------------------------*/

     #define CUDBG_IPC_FLAG_NAME                cudbgIpcFlag
     #define CUDBG_RPC_ENABLED                  cudbgRpcEnabled
     #define CUDBG_APICLIENT_PID                cudbgApiClientPid
     #define CUDBG_DEBUGGER_INITIALIZED         cudbgDebuggerInitialized
     #define CUDBG_APICLIENT_REVISION           cudbgApiClientRevision
     #define CUDBG_SESSION_ID                   cudbgSessionId
     #define CUDBG_ATTACH_HANDLER_AVAILABLE     cudbgAttachHandlerAvailable
     #define CUDBG_DETACH_SUSPENDED_DEVICES_MASK
cudbgDetachSuspendedDevicesMask
     #define CUDBG_ENABLE_LAUNCH_BLOCKING       cudbgEnableLaunchBlocking
     #define CUDBG_ENABLE_INTEGRATED_MEMCHECK   cudbgEnableIntegratedMemcheck
     #define CUDBG_ENABLE_PREEMPTION_DEBUGGING  cudbgEnablePreemptionDebugging

     /*---------------- Internal Breakpoint Entries for Error Reporting
------------*/

     #define CUDBG_REPORT_DRIVER_API_ERROR
cudbgReportDriverApiError
     #define CUDBG_REPORTED_DRIVER_API_ERROR_CODE
cudbgReportedDriverApiErrorCode
     #define CUDBG_REPORTED_DRIVER_API_ERROR_FUNC_NAME_SIZE
cudbgReportedDriverApiErrorFuncNameSize
     #define CUDBG_REPORTED_DRIVER_API_ERROR_FUNC_NAME_ADDR
cudbgReportedDriverApiErrorFuncNameAddr
     #define CUDBG_REPORT_DRIVER_INTERNAL_ERROR
cudbgReportDriverInternalError
     #define CUDBG_REPORTED_DRIVER_INTERNAL_ERROR_CODE
cudbgReportedDriverInternalErrorCode
```

```
     /*---------------------------- API Return Types
-----------------------------*/

     typedef enum {
         CUDBG_SUCCESS                            = 0x0000,  /* Successful
execution */
         CUDBG_ERROR_UNKNOWN                      = 0x0001,  /* Error type not
listed below */
         CUDBG_ERROR_BUFFER_TOO_SMALL             = 0x0002,  /* Cannot copy all
the queried data into the buffer argument */
         CUDBG_ERROR_UNKNOWN_FUNCTION             = 0x0003,  /* Function cannot
be found in the CUDA kernel */
         CUDBG_ERROR_INVALID_ARGS                 = 0x0004,  /* Wrong use of
arguments (NULL pointer, illegal value,...) */
         CUDBG_ERROR_UNINITIALIZED                = 0x0005,  /* Debugger API has
not yet been properly initialized */
         CUDBG_ERROR_INVALID_COORDINATES          = 0x0006,  /* Invalid block or
thread coordinates were provided */
         CUDBG_ERROR_INVALID_MEMORY_SEGMENT       = 0x0007,  /* Invalid memory
segment requested (read/write) */
         CUDBG_ERROR_INVALID_MEMORY_ACCESS        = 0x0008,  /* Requested
address (+size) is not within proper segment boundaries */
         CUDBG_ERROR_MEMORY_MAPPING_FAILED        = 0x0009,  /* Memory is not
mapped and can't be mapped */
         CUDBG_ERROR_INTERNAL                     = 0x000a,  /* A debugger
internal error occurred */
         CUDBG_ERROR_INVALID_DEVICE               = 0x000b,  /* Specified device
cannot be found */
         CUDBG_ERROR_INVALID_SM                   = 0x000c,  /* Specified sm
cannot be found */
         CUDBG_ERROR_INVALID_WARP                 = 0x000d,  /* Specified warp
cannot be found */
         CUDBG_ERROR_INVALID_LANE                 = 0x000e,  /* Specified lane
cannot be found */
         CUDBG_ERROR_SUSPENDED_DEVICE             = 0x000f,  /* device is
suspended */
         CUDBG_ERROR_RUNNING_DEVICE               = 0x0010,  /* device is
running and not suspended */
         CUDBG_ERROR_INVALID_ADDRESS              = 0x0012,  /* address is out-
of-range */
         CUDBG_ERROR_INCOMPATIBLE_API             = 0x0013,  /* API version does
not match */
         CUDBG_ERROR_INITIALIZATION_FAILURE       = 0x0014,  /* The CUDA Driver
failed to initialize */
         CUDBG_ERROR_INVALID_GRID                 = 0x0015,  /* Specified grid
cannot be found */
         CUDBG_ERROR_NO_EVENT_AVAILABLE           = 0x0016,  /* No event left to
be processed */
         CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED     = 0x0017,  /* One or more
devices have an associated watchdog (eg. X) */
         CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED      = 0x0018,  /* All devices have
an associated watchdog (eg. X) */
         CUDBG_ERROR_INVALID_ATTRIBUTE            = 0x0019,  /* Specified
attribute does not exist or is incorrect */
         CUDBG_ERROR_ZERO_CALL_DEPTH              = 0x001a,  /* No function
calls have been made on the device */
         CUDBG_ERROR_INVALID_CALL_LEVEL           = 0x001b,  /* Specified call
level is invalid */
         CUDBG_ERROR_COMMUNICATION_FAILURE        = 0x001c,  /* Communication
error between the debugger and the application. */
         CUDBG_ERROR_INVALID_CONTEXT              = 0x001d,  /* Specified
context cannot be found */
         CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM    = 0x001e,  /* Requested
address was not originally allocated from device memory (most likely visible in
system memory) */
         CUDBG_ERROR_MEMORY_UNMAPPING_FAILED      = 0x001f,  /* Memory is not
unmapped and can't be unmapped */
```

```
        CUDBG_ERROR_INCOMPATIBLE_DISPLAY_DRIVER = 0x0020,  /* The display
driver is incompatible with the API */
        CUDBG_ERROR_INVALID_MODULE              = 0x0021,  /* The specified
module is not valid */
        CUDBG_ERROR_LANE_NOT_IN_SYSCALL         = 0x0022,  /* The specified
lane is not inside a device syscall */
        CUDBG_ERROR_MEMCHECK_NOT_ENABLED        = 0x0023,  /* Memcheck has not
been enabled */
        CUDBG_ERROR_INVALID_ENVVAR_ARGS         = 0x0024,  /* Some environment
variable's value is invalid */
        CUDBG_ERROR_OS_RESOURCES                = 0x0025,  /* Error while
allocating resources from the OS */
        CUDBG_ERROR_FORK_FAILED                 = 0x0026,  /* Error while
forking the debugger process */
        CUDBG_ERROR_NO_DEVICE_AVAILABLE         = 0x0027,  /* No CUDA capable
device was found */
        CUDBG_ERROR_ATTACH_NOT_POSSIBLE         = 0x0028,  /* Attaching to the
CUDA program is not possible */
    } CUDBGResult;

    /*------------------------------ Grid Attributes
------------------------------*/

    typedef enum {
        CUDBG_ATTR_GRID_LAUNCH_BLOCKING    = 0x000,   /* Whether the grid
launch is blocking or not. */
        CUDBG_ATTR_GRID_TID                = 0x001,   /* Id of the host thread
that launched the grid. */
    } CUDBGAttribute;

    typedef struct {
        CUDBGAttribute attribute;
        uint64_t       value;
    } CUDBGAttributeValuePair;

    typedef enum {
        CUDBG_GRID_STATUS_INVALID,          /* An invalid grid ID was passed,
or an error occurred during status lookup */
        CUDBG_GRID_STATUS_PENDING,          /* The grid was launched but is
not running on the HW yet */
        CUDBG_GRID_STATUS_ACTIVE,           /* The grid is currently running
on the HW */
        CUDBG_GRID_STATUS_SLEEPING,         /* The grid is on the device,
doing a join */
        CUDBG_GRID_STATUS_TERMINATED,       /* The grid has finished executing
*/
        CUDBG_GRID_STATUS_UNDETERMINED,     /* The grid is either PENDING or
TERMINATED */
    } CUDBGGridStatus;

    /*------------------------------ Kernel Types
------------------------------*/

    typedef enum {
        CUDBG_KNL_TYPE_UNKNOWN             = 0x000,   /* Any type not listed
below. */
        CUDBG_KNL_TYPE_SYSTEM              = 0x001,   /* System kernel, such
as MemCpy. */
        CUDBG_KNL_TYPE_APPLICATION         = 0x002,   /* Application kernel,
user-defined or libraries. */
    } CUDBGKernelType;

    /*------------------------- Physical Register Types
-------------------------*/

    typedef enum {
        REG_CLASS_INVALID                  = 0x000,   /* invalid register */
```

```
        REG_CLASS_REG_CC                    = 0x001,    /* Condition register */
        REG_CLASS_REG_PRED                  = 0x002,    /* Predicate register */
        REG_CLASS_REG_ADDR                  = 0x003,    /* Address register */
        REG_CLASS_REG_HALF                  = 0x004,    /* 16-bit register
(Currently unused) */
        REG_CLASS_REG_FULL                  = 0x005,    /* 32-bit register */
        REG_CLASS_MEM_LOCAL                 = 0x006,    /* register spilled in
memory */
        REG_CLASS_LMEM_REG_OFFSET           = 0x007,    /* register at stack
offset (ABI only) */
    } CUDBGRegClass;

    /*---------------------------- Application Events
----------------------------*/

    typedef enum {
        CUDBG_EVENT_INVALID                 = 0x000,    /* Invalid event */
        CUDBG_EVENT_ELF_IMAGE_LOADED        = 0x001,    /* ELF image for CUDA
kernel(s) is ready */
        CUDBG_EVENT_KERNEL_READY            = 0x002,    /* A CUDA kernel is
ready to be launched */
        CUDBG_EVENT_KERNEL_FINISHED         = 0x003,    /* A CUDA kernel has
terminated */
        CUDBG_EVENT_INTERNAL_ERROR          = 0x004,    /* Unexpected error. The
API may be unstable. */
        CUDBG_EVENT_CTX_PUSH                = 0x005,    /* A CUDA context has
been pushed. */
        CUDBG_EVENT_CTX_POP                 = 0x006,    /* A CUDA context has
been popped. */
        CUDBG_EVENT_CTX_CREATE              = 0x007,    /* A CUDA context has
been created and pushed. */
        CUDBG_EVENT_CTX_DESTROY             = 0x008,    /* A CUDA context has
been, popped if pushed, then destroyed. */
        CUDBG_EVENT_TIMEOUT                 = 0x009,    /* Nothing happened for
a while. This is heartbeat event. */
        CUDBG_EVENT_ATTACH_COMPLETE         = 0x00a,    /* Attach complete. */
        CUDBG_EVENT_DETACH_COMPLETE         = 0x00b,    /* Detach complete. */
    } CUDBGEventKind;

    /*------------------------------ Kernel Origin
------------------------------*/

    typedef enum {
        CUDBG_KNL_ORIGIN_CPU                = 0x000,    /* The kernel was
launched from the CPU. */
        CUDBG_KNL_ORIGIN_GPU                = 0x001,    /* The kernel was
launched from the GPU. */
    } CUDBGKernelOrigin;

    /*------------------------ Kernel Launch Notify Mode
------------------------*/

    typedef enum {
        CUDBG_KNL_LAUNCH_NOTIFY_EVENT       = 0x000,    /* The kernel
notifications generate events */
        CUDBG_KNL_LAUNCH_NOTIFY_DEFER       = 0x001,    /* The kernel
notifications are deferred */
    } CUDBGKernelLaunchNotifyMode;

    /*------------------------------ Code Address
-------------------------------*/

    /* Deprecated */
    typedef struct {
        CUDBGEventKind kind;
        union cases30_st {
            struct elfImageLoaded30_st {
```

```
                char     *relocatedElfImage;
                char     *nonRelocatedElfImage;
                uint32_t  size;
            } elfImageLoaded;
            struct kernelReady30_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
            } kernelReady;
            struct kernelFinished30_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
            } kernelFinished;
        } cases;
    } CUDBGEvent30;

    /* Deprecated */
    typedef struct {
        CUDBGEventKind kind;
        union cases32_st {
            struct elfImageLoaded32_st {
                char     *relocatedElfImage;
                char     *nonRelocatedElfImage;
                uint32_t  size;
                uint32_t  dev;
                uint64_t  context;
                uint64_t  module;
            } elfImageLoaded;
            struct kernelReady32_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
            } kernelReady;
            struct kernelFinished32_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
            } kernelFinished;
            struct contextPush32_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPush;
            struct contextPop32_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPop;
            struct contextCreate32_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextCreate;
            struct contextDestroy32_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextDestroy;
```

```
        } cases;
    } CUDBGEvent32;

    /* Deprecated */
    typedef struct {
        CUDBGEventKind kind;
        union cases42_st {
            struct elfImageLoaded42_st {
                char     *relocatedElfImage;
                char     *nonRelocatedElfImage;
                uint32_t  size32;
                uint32_t  dev;
                uint64_t  context;
                uint64_t  module;
                uint64_t  size;
            } elfImageLoaded;
            struct kernelReady42_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
                CuDim3   gridDim;
                CuDim3   blockDim;
                CUDBGKernelType type;
            } kernelReady;
            struct kernelFinished42_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
            } kernelFinished;
            struct contextPush42_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPush;
            struct contextPop42_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPop;
            struct contextCreate42_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextCreate;
            struct contextDestroy42_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextDestroy;
        } cases;
    } CUDBGEvent42;

    typedef struct {
        CUDBGEventKind kind;
        union cases50_st {
            struct elfImageLoaded50_st {
                char     *relocatedElfImage;
                char     *nonRelocatedElfImage;
                uint32_t  size32;
```

```
                uint32_t  dev;
                uint64_t  context;
                uint64_t  module;
                uint64_t  size;
            } elfImageLoaded;
            struct kernelReady50_st{
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
                CuDim3   gridDim;
                CuDim3   blockDim;
                CUDBGKernelType type;
            } kernelReady;
            struct kernelFinished50_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
            } kernelFinished;
            struct contextPush50_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPush;
            struct contextPop50_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPop;
            struct contextCreate50_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextCreate;
            struct contextDestroy50_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextDestroy;
            struct internalError50_st {
                CUDBGResult errorType;
            } internalError;
        } cases;
    } CUDBGEvent50;

    typedef struct {
        CUDBGEventKind kind;
        union cases_st {
            struct elfImageLoaded_st {
                char     *relocatedElfImage;
                char     *nonRelocatedElfImage;
                uint32_t  size32;
                uint32_t  dev;
                uint64_t  context;
                uint64_t  module;
                uint64_t  size;
            } elfImageLoaded;
            struct kernelReady_st{
                uint32_t dev;
                uint32_t gridId;
```

```
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
                CuDim3   gridDim;
                CuDim3   blockDim;
                CUDBGKernelType type;
                uint64_t parentGridId;
                uint64_t gridId64;
                CUDBGKernelOrigin origin;
            } kernelReady;
            struct kernelFinished_st {
                uint32_t dev;
                uint32_t gridId;
                uint32_t tid;
                uint64_t context;
                uint64_t module;
                uint64_t function;
                uint64_t functionEntry;
                uint64_t gridId64;
            } kernelFinished;
            struct contextPush_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPush;
            struct contextPop_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextPop;
            struct contextCreate_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextCreate;
            struct contextDestroy_st {
                uint32_t dev;
                uint32_t tid;
                uint64_t context;
            } contextDestroy;
            struct internalError_st {
                CUDBGResult errorType;
            } internalError;
        } cases;
    } CUDBGEvent;

    typedef struct {
        uint32_t tid;
    } CUDBGEventCallbackData40;

    typedef struct {
        uint32_t tid;
        uint32_t timeout;
    } CUDBGEventCallbackData;

    #pragma pack(push,1)
    typedef struct {
        uint32_t dev;
        uint64_t gridId64;
        uint32_t tid;
        uint64_t context;
        uint64_t module;
        uint64_t function;
        uint64_t functionEntry;
        CuDim3   gridDim;
```

```
        CuDim3   blockDim;
        CUDBGKernelType type;
        uint64_t parentGridId;
        CUDBGKernelOrigin origin;
    } CUDBGGridInfo;
    #pragma pack(pop)

    typedef void (*CUDBGNotifyNewEventCallback31)(void *data);
    typedef void (*CUDBGNotifyNewEventCallback40)(CUDBGEventCallbackData40
*data);
    typedef void (*CUDBGNotifyNewEventCallback)(CUDBGEventCallbackData *data);

    /*---------------------------------- Exceptions
-----------------------------*/

    typedef enum {
        CUDBG_EXCEPTION_UNKNOWN = 0xFFFFFFFFU, // Force
sizeof(CUDBGException_t)==4
        CUDBG_EXCEPTION_NONE = 0,
        CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS = 1,
        CUDBG_EXCEPTION_LANE_USER_STACK_OVERFLOW = 2,
        CUDBG_EXCEPTION_DEVICE_HARDWARE_STACK_OVERFLOW = 3,
        CUDBG_EXCEPTION_WARP_ILLEGAL_INSTRUCTION = 4,
        CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS = 5,
        CUDBG_EXCEPTION_WARP_MISALIGNED_ADDRESS = 6,
        CUDBG_EXCEPTION_WARP_INVALID_ADDRESS_SPACE = 7,
        CUDBG_EXCEPTION_WARP_INVALID_PC = 8,
        CUDBG_EXCEPTION_WARP_HARDWARE_STACK_OVERFLOW = 9,
        CUDBG_EXCEPTION_DEVICE_ILLEGAL_ADDRESS = 10,
        CUDBG_EXCEPTION_LANE_MISALIGNED_ADDRESS = 11,
        CUDBG_EXCEPTION_WARP_ASSERT = 12,
        CUDBG_EXCEPTION_LANE_SYSCALL_ERROR = 13,
        CUDBG_EXCEPTION_WARP_ILLEGAL_ADDRESS = 14,
    } CUDBGException_t;

    /*---------------------------------- Exports
-----------------------------*/

    typedef const struct CUDBGAPI_st *CUDBGAPI;

    CUDBGResult cudbgGetAPI(uint32_t major, uint32_t minor, uint32_t rev,
CUDBGAPI *api);
    CUDBGResult cudbgGetAPIVersion(uint32_t *major, uint32_t *minor, uint32_t
*rev);
    CUDBGResult cudbgMain(int apiClientPid, uint32_t apiClientRevision, int
sessionId, int attachState,
                          int attachEventInitialized, int writeFd, int
detachFd, int attachStubInUse,
                          int enablePreemptionDebugging);

    struct CUDBGAPI_st {
        /* Initialization */
        CUDBGResult (*initialize)(void);
        CUDBGResult (*finalize)(void);

        /* Device Execution Control */
        CUDBGResult (*suspendDevice)(uint32_t dev);
        CUDBGResult (*resumeDevice)(uint32_t dev);
        CUDBGResult (*singleStepWarp40)(uint32_t dev, uint32_t sm, uint32_t
wp);

        /* Breakpoints */
        CUDBGResult (*setBreakpoint31)(uint64_t addr);
        CUDBGResult (*unsetBreakpoint31)(uint64_t addr);

        /* Device State Inspection */
```

```
        CUDBGResult (*readGridId50)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t *gridId);
        CUDBGResult (*readBlockIdx32)(uint32_t dev, uint32_t sm, uint32_t wp,
CuDim2 *blockIdx);
        CUDBGResult (*readThreadIdx)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, CuDim3 *threadIdx);
        CUDBGResult (*readBrokenWarps)(uint32_t dev, uint32_t sm, uint64_t
*brokenWarpsMask);
        CUDBGResult (*readValidWarps)(uint32_t dev, uint32_t sm, uint64_t
*validWarpsMask);
        CUDBGResult (*readValidLanes)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t *validLanesMask);
        CUDBGResult (*readActiveLanes)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t *activeLanesMask);
        CUDBGResult (*readCodeMemory)(uint32_t dev, uint64_t addr, void *buf,
uint32_t sz);
        CUDBGResult (*readConstMemory)(uint32_t dev, uint64_t addr, void *buf,
uint32_t sz);
        CUDBGResult (*readGlobalMemory31)(uint32_t dev, uint64_t addr, void
*buf, uint32_t sz);
        CUDBGResult (*readParamMemory)(uint32_t dev, uint32_t sm, uint32_t wp,
uint64_t addr, void *buf, uint32_t sz);
        CUDBGResult (*readSharedMemory)(uint32_t dev, uint32_t sm, uint32_t
wp, uint64_t addr, void *buf, uint32_t sz);
        CUDBGResult (*readLocalMemory)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, uint64_t addr, void *buf, uint32_t sz);
        CUDBGResult (*readRegister)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, uint32_t regno, uint32_t *val);
        CUDBGResult (*readPC)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t
ln, uint64_t *pc);
        CUDBGResult (*readVirtualPC)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, uint64_t *pc);
        CUDBGResult (*readLaneStatus)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, bool *error);

        /* Device State Alteration */
        CUDBGResult (*writeGlobalMemory31)(uint32_t dev, uint64_t addr, const
void *buf, uint32_t sz);
        CUDBGResult (*writeParamMemory)(uint32_t dev, uint32_t sm, uint32_t
wp, uint64_t addr, const void *buf, uint32_t sz);
        CUDBGResult (*writeSharedMemory)(uint32_t dev, uint32_t sm, uint32_t
wp, uint64_t addr, const void *buf, uint32_t sz);
        CUDBGResult (*writeLocalMemory)(uint32_t dev, uint32_t sm, uint32_t
wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz);
        CUDBGResult (*writeRegister)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, uint32_t regno, uint32_t val);

        /* Grid Properties */
        CUDBGResult (*getGridDim32)(uint32_t dev, uint32_t sm, uint32_t wp,
CuDim2 *gridDim);
        CUDBGResult (*getBlockDim)(uint32_t dev, uint32_t sm, uint32_t wp,
CuDim3 *blockDim);
        CUDBGResult (*getTID)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t
*tid);
        CUDBGResult (*getElfImage32)(uint32_t dev, uint32_t sm, uint32_t wp,
bool relocated, void **elfImage, uint32_t *size);

        /* Device Properties */
        CUDBGResult (*getDeviceType)(uint32_t dev, char *buf, uint32_t sz);
        CUDBGResult (*getSmType)(uint32_t dev, char *buf, uint32_t sz);
        CUDBGResult (*getNumDevices)(uint32_t *numDev);
        CUDBGResult (*getNumSMs)(uint32_t dev, uint32_t *numSMs);
        CUDBGResult (*getNumWarps)(uint32_t dev, uint32_t *numWarps);
        CUDBGResult (*getNumLanes)(uint32_t dev, uint32_t *numLanes);
        CUDBGResult (*getNumRegisters)(uint32_t dev, uint32_t *numRegs);

        /* DWARF-related routines */
```

```
        CUDBGResult (*getPhysicalRegister30)(uint64_t pc, char *reg, uint32_t
*buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass);
        CUDBGResult (*disassemble)(uint32_t dev, uint64_t addr, uint32_t
*instSize, char *buf, uint32_t sz);
        CUDBGResult (*isDeviceCodeAddress)(uintptr_t addr, bool
*isDeviceAddress);
        CUDBGResult (*lookupDeviceCodeSymbol)(char *symName, bool *symFound,
uintptr_t *symAddr);

        /* Events */
        CUDBGResult (*setNotifyNewEventCallback31)
(CUDBGNotifyNewEventCallback31 callback, void *data);
        CUDBGResult (*getNextEvent30)(CUDBGEvent30 *event);
        CUDBGResult (*acknowledgeEvent30)(CUDBGEvent30 *event);

        /* 3.1 Extensions */
        CUDBGResult (*getGridAttribute)(uint32_t dev, uint32_t sm, uint32_t
wp, CUDBGAttribute attr, uint64_t *value);
        CUDBGResult (*getGridAttributes)(uint32_t dev, uint32_t sm, uint32_t
wp, CUDBGAttributeValuePair *pairs, uint32_t numPairs);
        CUDBGResult (*getPhysicalRegister40)(uint32_t dev, uint32_t sm,
uint32_t wp, uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t
*numPhysRegs, CUDBGRegClass *regClass);
        CUDBGResult (*readLaneException)(uint32_t dev, uint32_t sm, uint32_t
wp, uint32_t ln, CUDBGException_t *exception);
        CUDBGResult (*getNextEvent32)(CUDBGEvent32 *event);
        CUDBGResult (*acknowledgeEvents42)(void);

        /* 3.1 - ABI */
        CUDBGResult (*readCallDepth32)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t *depth);
        CUDBGResult (*readReturnAddress32)(uint32_t dev, uint32_t sm, uint32_t
wp, uint32_t level, uint64_t *ra);
        CUDBGResult (*readVirtualReturnAddress32)(uint32_t dev, uint32_t sm,
uint32_t wp, uint32_t level, uint64_t *ra);

        /* 3.2 Extensions */
        CUDBGResult (*readGlobalMemory)(uint32_t dev, uint32_t sm, uint32_t
wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz);
        CUDBGResult (*writeGlobalMemory)(uint32_t dev, uint32_t sm, uint32_t
wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz);
        CUDBGResult (*readPinnedMemory)(uint64_t addr, void *buf, uint32_t
sz);
        CUDBGResult (*writePinnedMemory)(uint64_t addr, const void *buf,
uint32_t sz);
        CUDBGResult (*setBreakpoint)(uint32_t dev, uint64_t addr);
        CUDBGResult (*unsetBreakpoint)(uint32_t dev, uint64_t addr);
        CUDBGResult (*setNotifyNewEventCallback40)
(CUDBGNotifyNewEventCallback40 callback);

        /* 4.0 Extensions */
        CUDBGResult (*getNextEvent42)(CUDBGEvent42 *event);
        CUDBGResult (*readTextureMemory)(uint32_t devId, uint32_t vsm,
uint32_t wp, uint32_t id, uint32_t dim, uint32_t *coords, void *buf, uint32_t
sz);
        CUDBGResult (*readBlockIdx)(uint32_t dev, uint32_t sm, uint32_t wp,
CuDim3 *blockIdx);
        CUDBGResult (*getGridDim)(uint32_t dev, uint32_t sm, uint32_t wp,
CuDim3 *gridDim);
        CUDBGResult (*readCallDepth)(uint32_t dev, uint32_t sm, uint32_t wp,
uint32_t ln, uint32_t *depth);
        CUDBGResult (*readReturnAddress)(uint32_t dev, uint32_t sm, uint32_t
wp, uint32_t ln, uint32_t level, uint64_t *ra);
        CUDBGResult (*readVirtualReturnAddress)(uint32_t dev, uint32_t sm,
uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra);
        CUDBGResult (*getElfImage)(uint32_t dev, uint32_t sm, uint32_t wp,
bool relocated, void **elfImage, uint64_t *size);
```

```
        /* 4.1 Extensions */
        CUDBGResult (*getHostAddrFromDeviceAddr)(uint32_t dev, uint64_t
device_addr, uint64_t *host_addr);
        CUDBGResult (*singleStepWarp)(uint32_t dev, uint32_t sm, uint32_t wp,
uint64_t *warpMask);
        CUDBGResult (*setNotifyNewEventCallback)(CUDBGNotifyNewEventCallback
callback);
        CUDBGResult (*readSyscallCallDepth)(uint32_t dev, uint32_t sm,
uint32_t wp, uint32_t ln, uint32_t *depth);

        /* 4.2 Extensions */
        CUDBGResult (*readTextureMemoryBindless)(uint32_t devId, uint32_t vsm,
uint32_t wp, uint32_t texSymtabIndex, uint32_t dim, uint32_t *coords, void
*buf, uint32_t sz);

        /* 5.0 Extensions */
        CUDBGResult (*clearAttachState)(void);
        CUDBGResult (*getNextSyncEvent50)(CUDBGEvent50 *event);
        CUDBGResult (*memcheckReadErrorAddress)(uint32_t dev, uint32_t sm,
uint32_t wp, uint32_t ln, uint64_t *address, ptxStorageKind *storage);
        CUDBGResult (*acknowledgeSyncEvents)(void);
        CUDBGResult (*getNextAsyncEvent50)(CUDBGEvent50 *event);
        CUDBGResult (*requestCleanupOnDetach)(void);
        CUDBGResult (*initializeAttachStub)(void);
        CUDBGResult (*getGridStatus50)(uint32_t dev, uint32_t
gridId, CUDBGGridStatus *status);

        /* 5.5 Extensions */
        CUDBGResult (*getNextSyncEvent)(CUDBGEvent *event);
        CUDBGResult (*getNextAsyncEvent)(CUDBGEvent *event);
        CUDBGResult (*getGridInfo)(uint32_t dev, uint64_t
gridId64, CUDBGGridInfo *gridInfo);
        CUDBGResult (*readGridId)(uint32_t dev, uint32_t sm, uint32_t wp,
uint64_t *gridId64);
        CUDBGResult (*getGridStatus)(uint32_t dev, uint64_t
gridId64, CUDBGGridStatus *status);
        CUDBGResult (*setKernelLaunchNotificationMode)
(CUDBGKernelLaunchNotifyMode mode);
        CUDBGResult (*getDevicePCIBusInfo) (uint32_t devId, uint32_t
*pciBusId, uint32_t *pciDevId);
        CUDBGResult (*readDeviceExceptionState) (uint32_t devId, uint64_t
*exceptionSMMask);
    };

    #ifdef __cplusplus
    }
    #endif

    #endif
```

struct CUDBGAPI_st

The CUDA debugger API routines.

struct CUDBGEvent

Event information container.

struct CUDBGEventCallbackData

Event information passed to callback set with setNotifyNewEventCallback function.

struct CUDBGEventCallbackData40

Event information passed to callback set with setNotifyNewEventCallback function.

struct CUDBGGridInfo

Grid info.

Chapter 6. GLOBALS

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

CUDBG_ATTR_GRID_LAUNCH_BLOCKING
cudadebugger.h
CUDBG_ATTR_GRID_TID
cudadebugger.h
CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM
cudadebugger.h
CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED
cudadebugger.h
CUDBG_ERROR_ATTACH_NOT_POSSIBLE
cudadebugger.h
CUDBG_ERROR_BUFFER_TOO_SMALL
cudadebugger.h
CUDBG_ERROR_COMMUNICATION_FAILURE
cudadebugger.h
CUDBG_ERROR_FORK_FAILED
cudadebugger.h
CUDBG_ERROR_INCOMPATIBLE_API
cudadebugger.h
CUDBG_ERROR_INCOMPATIBLE_DISPLAY_DRIVER
cudadebugger.h
CUDBG_ERROR_INITIALIZATION_FAILURE
cudadebugger.h
CUDBG_ERROR_INTERNAL
cudadebugger.h
CUDBG_ERROR_INVALID_ADDRESS
cudadebugger.h
CUDBG_ERROR_INVALID_ARGS
cudadebugger.h

CUDBG_ERROR_INVALID_ATTRIBUTE
cudadebugger.h
CUDBG_ERROR_INVALID_CALL_LEVEL
cudadebugger.h
CUDBG_ERROR_INVALID_CONTEXT
cudadebugger.h
CUDBG_ERROR_INVALID_COORDINATES
cudadebugger.h
CUDBG_ERROR_INVALID_DEVICE
cudadebugger.h
CUDBG_ERROR_INVALID_ENVVAR_ARGS
cudadebugger.h
CUDBG_ERROR_INVALID_GRID
cudadebugger.h
CUDBG_ERROR_INVALID_LANE
cudadebugger.h
CUDBG_ERROR_INVALID_MEMORY_ACCESS
cudadebugger.h
CUDBG_ERROR_INVALID_MEMORY_SEGMENT
cudadebugger.h
CUDBG_ERROR_INVALID_MODULE
cudadebugger.h
CUDBG_ERROR_INVALID_SM
cudadebugger.h
CUDBG_ERROR_INVALID_WARP
cudadebugger.h
CUDBG_ERROR_LANE_NOT_IN_SYSCALL
cudadebugger.h
CUDBG_ERROR_MEMCHECK_NOT_ENABLED
cudadebugger.h
CUDBG_ERROR_MEMORY_MAPPING_FAILED
cudadebugger.h
CUDBG_ERROR_MEMORY_UNMAPPING_FAILED
cudadebugger.h
CUDBG_ERROR_NO_DEVICE_AVAILABLE
cudadebugger.h
CUDBG_ERROR_NO_EVENT_AVAILABLE
cudadebugger.h
CUDBG_ERROR_OS_RESOURCES
cudadebugger.h
CUDBG_ERROR_RUNNING_DEVICE
cudadebugger.h

CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED
cudadebugger.h
CUDBG_ERROR_SUSPENDED_DEVICE
cudadebugger.h
CUDBG_ERROR_UNINITIALIZED
cudadebugger.h
CUDBG_ERROR_UNKNOWN
cudadebugger.h
CUDBG_ERROR_UNKNOWN_FUNCTION
cudadebugger.h
CUDBG_ERROR_ZERO_CALL_DEPTH
cudadebugger.h
CUDBG_EVENT_ATTACH_COMPLETE
cudadebugger.h
CUDBG_EVENT_CTX_CREATE
cudadebugger.h
CUDBG_EVENT_CTX_DESTROY
cudadebugger.h
CUDBG_EVENT_CTX_POP
cudadebugger.h
CUDBG_EVENT_CTX_PUSH
cudadebugger.h
CUDBG_EVENT_ELF_IMAGE_LOADED
cudadebugger.h
CUDBG_EVENT_INTERNAL_ERROR
cudadebugger.h
CUDBG_EVENT_INVALID
cudadebugger.h
CUDBG_EVENT_KERNEL_FINISHED
cudadebugger.h
CUDBG_EVENT_KERNEL_READY
cudadebugger.h
CUDBG_EVENT_TIMEOUT
cudadebugger.h
CUDBG_EXCEPTION_DEVICE_HARDWARE_STACK_OVERFLOW
cudadebugger.h
CUDBG_EXCEPTION_DEVICE_ILLEGAL_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_LANE_MISALIGNED_ADDRESS
cudadebugger.h

CUDBG_EXCEPTION_LANE_USER_STACK_OVERFLOW
cudadebugger.h
CUDBG_EXCEPTION_NONE
cudadebugger.h
CUDBG_EXCEPTION_UNKNOWN
cudadebugger.h
CUDBG_EXCEPTION_WARP_HARDWARE_STACK_OVERFLOW
cudadebugger.h
CUDBG_EXCEPTION_WARP_ILLEGAL_INSTRUCTION
cudadebugger.h
CUDBG_EXCEPTION_WARP_INVALID_ADDRESS_SPACE
cudadebugger.h
CUDBG_EXCEPTION_WARP_INVALID_PC
cudadebugger.h
CUDBG_EXCEPTION_WARP_MISALIGNED_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS
cudadebugger.h
CUDBG_GRID_STATUS_ACTIVE
cudadebugger.h
CUDBG_GRID_STATUS_INVALID
cudadebugger.h
CUDBG_GRID_STATUS_PENDING
cudadebugger.h
CUDBG_GRID_STATUS_SLEEPING
cudadebugger.h
CUDBG_GRID_STATUS_TERMINATED
cudadebugger.h
CUDBG_GRID_STATUS_UNDETERMINED
cudadebugger.h
CUDBG_KNL_LAUNCH_NOTIFY_EVENT
cudadebugger.h
CUDBG_KNL_ORIGIN_CPU
cudadebugger.h
CUDBG_KNL_ORIGIN_GPU
cudadebugger.h
CUDBG_KNL_TYPE_APPLICATION
cudadebugger.h
CUDBG_KNL_TYPE_SYSTEM
cudadebugger.h
CUDBG_KNL_TYPE_UNKNOWN
cudadebugger.h

CUDBG_SUCCESS
cudadebugger.h
CUDBGAttribute
cudadebugger.h
CUDBGEventKind
cudadebugger.h
CUDBGException_t
cudadebugger.h
cudbgGetAPIVersion()
cudadebugger.h
CUDBGGridStatus
cudadebugger.h
CUDBGKernelLaunchNotifyMode
cudadebugger.h
CUDBGKernelOrigin
cudadebugger.h
CUDBGKernelType
cudadebugger.h
CUDBGNotifyNewEventCallback
cudadebugger.h
CUDBGNotifyNewEventCallback31
cudadebugger.h
CUDBGRegClass
cudadebugger.h
CUDBGResult
cudadebugger.h

REG_CLASS_INVALID
cudadebugger.h
REG_CLASS_LMEM_REG_OFFSET
cudadebugger.h
REG_CLASS_MEM_LOCAL
cudadebugger.h
REG_CLASS_REG_ADDR
cudadebugger.h
REG_CLASS_REG_CC
cudadebugger.h
REG_CLASS_REG_FULL
cudadebugger.h
REG_CLASS_REG_HALF
cudadebugger.h
REG_CLASS_REG_PRED
cudadebugger.h

6.1. Globals - Functions

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

cudbgGetAPIVersion()
cudadebugger.h

6.2. Globals - Typedefs

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

CUDBGNotifyNewEventCallback
cudadebugger.h
CUDBGNotifyNewEventCallback31
cudadebugger.h

6.3. Globals - Enumerations

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

CUDBGAttribute
cudadebugger.h
CUDBGEventKind
cudadebugger.h
CUDBGException_t
cudadebugger.h
CUDBGGridStatus
cudadebugger.h
CUDBGKernelLaunchNotifyMode
cudadebugger.h
CUDBGKernelOrigin
cudadebugger.h
CUDBGKernelType
cudadebugger.h
CUDBGRegClass
cudadebugger.h
CUDBGResult
cudadebugger.h

6.4. Globals - Enumerator

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

CUDBG_ATTR_GRID_LAUNCH_BLOCKING
cudadebugger.h
CUDBG_ATTR_GRID_TID
cudadebugger.h
CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM
cudadebugger.h
CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED
cudadebugger.h
CUDBG_ERROR_ATTACH_NOT_POSSIBLE
cudadebugger.h
CUDBG_ERROR_BUFFER_TOO_SMALL
cudadebugger.h
CUDBG_ERROR_COMMUNICATION_FAILURE
cudadebugger.h
CUDBG_ERROR_FORK_FAILED
cudadebugger.h
CUDBG_ERROR_INCOMPATIBLE_API
cudadebugger.h
CUDBG_ERROR_INCOMPATIBLE_DISPLAY_DRIVER
cudadebugger.h
CUDBG_ERROR_INITIALIZATION_FAILURE
cudadebugger.h
CUDBG_ERROR_INTERNAL
cudadebugger.h
CUDBG_ERROR_INVALID_ADDRESS
cudadebugger.h
CUDBG_ERROR_INVALID_ARGS
cudadebugger.h
CUDBG_ERROR_INVALID_ATTRIBUTE
cudadebugger.h
CUDBG_ERROR_INVALID_CALL_LEVEL
cudadebugger.h
CUDBG_ERROR_INVALID_CONTEXT
cudadebugger.h
CUDBG_ERROR_INVALID_COORDINATES
cudadebugger.h

CUDBG_ERROR_INVALID_DEVICE
cudadebugger.h
CUDBG_ERROR_INVALID_ENVVAR_ARGS
cudadebugger.h
CUDBG_ERROR_INVALID_GRID
cudadebugger.h
CUDBG_ERROR_INVALID_LANE
cudadebugger.h
CUDBG_ERROR_INVALID_MEMORY_ACCESS
cudadebugger.h
CUDBG_ERROR_INVALID_MEMORY_SEGMENT
cudadebugger.h
CUDBG_ERROR_INVALID_MODULE
cudadebugger.h
CUDBG_ERROR_INVALID_SM
cudadebugger.h
CUDBG_ERROR_INVALID_WARP
cudadebugger.h
CUDBG_ERROR_LANE_NOT_IN_SYSCALL
cudadebugger.h
CUDBG_ERROR_MEMCHECK_NOT_ENABLED
cudadebugger.h
CUDBG_ERROR_MEMORY_MAPPING_FAILED
cudadebugger.h
CUDBG_ERROR_MEMORY_UNMAPPING_FAILED
cudadebugger.h
CUDBG_ERROR_NO_DEVICE_AVAILABLE
cudadebugger.h
CUDBG_ERROR_NO_EVENT_AVAILABLE
cudadebugger.h
CUDBG_ERROR_OS_RESOURCES
cudadebugger.h
CUDBG_ERROR_RUNNING_DEVICE
cudadebugger.h
CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED
cudadebugger.h
CUDBG_ERROR_SUSPENDED_DEVICE
cudadebugger.h
CUDBG_ERROR_UNINITIALIZED
cudadebugger.h
CUDBG_ERROR_UNKNOWN
cudadebugger.h

CUDBG_ERROR_UNKNOWN_FUNCTION
cudadebugger.h
CUDBG_ERROR_ZERO_CALL_DEPTH
cudadebugger.h
CUDBG_EVENT_ATTACH_COMPLETE
cudadebugger.h
CUDBG_EVENT_CTX_CREATE
cudadebugger.h
CUDBG_EVENT_CTX_DESTROY
cudadebugger.h
CUDBG_EVENT_CTX_POP
cudadebugger.h
CUDBG_EVENT_CTX_PUSH
cudadebugger.h
CUDBG_EVENT_ELF_IMAGE_LOADED
cudadebugger.h
CUDBG_EVENT_INTERNAL_ERROR
cudadebugger.h
CUDBG_EVENT_INVALID
cudadebugger.h
CUDBG_EVENT_KERNEL_FINISHED
cudadebugger.h
CUDBG_EVENT_KERNEL_READY
cudadebugger.h
CUDBG_EVENT_TIMEOUT
cudadebugger.h
CUDBG_EXCEPTION_DEVICE_HARDWARE_STACK_OVERFLOW
cudadebugger.h
CUDBG_EXCEPTION_DEVICE_ILLEGAL_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_LANE_MISALIGNED_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_LANE_USER_STACK_OVERFLOW
cudadebugger.h
CUDBG_EXCEPTION_NONE
cudadebugger.h
CUDBG_EXCEPTION_UNKNOWN
cudadebugger.h
CUDBG_EXCEPTION_WARP_HARDWARE_STACK_OVERFLOW
cudadebugger.h

CUDBG_EXCEPTION_WARP_ILLEGAL_INSTRUCTION
cudadebugger.h
CUDBG_EXCEPTION_WARP_INVALID_ADDRESS_SPACE
cudadebugger.h
CUDBG_EXCEPTION_WARP_INVALID_PC
cudadebugger.h
CUDBG_EXCEPTION_WARP_MISALIGNED_ADDRESS
cudadebugger.h
CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS
cudadebugger.h
CUDBG_GRID_STATUS_ACTIVE
cudadebugger.h
CUDBG_GRID_STATUS_INVALID
cudadebugger.h
CUDBG_GRID_STATUS_PENDING
cudadebugger.h
CUDBG_GRID_STATUS_SLEEPING
cudadebugger.h
CUDBG_GRID_STATUS_TERMINATED
cudadebugger.h
CUDBG_GRID_STATUS_UNDETERMINED
cudadebugger.h
CUDBG_KNL_LAUNCH_NOTIFY_EVENT
cudadebugger.h
CUDBG_KNL_ORIGIN_CPU
cudadebugger.h
CUDBG_KNL_ORIGIN_GPU
cudadebugger.h
CUDBG_KNL_TYPE_APPLICATION
cudadebugger.h
CUDBG_KNL_TYPE_SYSTEM
cudadebugger.h
CUDBG_KNL_TYPE_UNKNOWN
cudadebugger.h
CUDBG_SUCCESS
cudadebugger.h

REG_CLASS_INVALID
cudadebugger.h
REG_CLASS_LMEM_REG_OFFSET
cudadebugger.h
REG_CLASS_MEM_LOCAL
cudadebugger.h

REG_CLASS_REG_ADDR
cudadebugger.h
REG_CLASS_REG_CC
cudadebugger.h
REG_CLASS_REG_FULL
cudadebugger.h
REG_CLASS_REG_HALF
cudadebugger.h
REG_CLASS_REG_PRED
cudadebugger.h

Chapter 7. DEPRECATED LIST

Global CUDBGEvent::cases_st::elfImageLoaded_st::size32

in CUDA 4.0.

Class CUDBGEventCallbackData40

in CUDA 4.1.

Global CUDBGAPI_st::singleStepWarp40)(uint32_t dev, uint32_t sm, uint32_t wp)

in CUDA 4.1.

Global CUDBGAPI_st::setBreakpoint31)(uint64_t addr)

in CUDA 3.2.

Global CUDBGAPI_st::unsetBreakpoint31)(uint64_t addr)

in CUDA 3.2.

Global CUDBGAPI_st::readBlockIdx32)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *blockIdx)

in CUDA 4.0.

Global CUDBGAPI_st::readCallDepth32)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *depth)

in CUDA 4.0.

Global CUDBGAPI_st::readGlobalMemory31)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)

in CUDA 3.2.

Global CUDBGAPI_st::readGridId50)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)

in CUDA 5.5.

Global CUDBGAPI_st::readReturnAddress32)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)

in CUDA 4.0.

Global CUDBGAPI_st::readVirtualReturnAddress32)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)

in CUDA 4.0.

Global CUDBGAPI_st::writeGlobalMemory31)(uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)

in CUDA 3.2.

Global CUDBGAPI_st::getElfImage32)(uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint32_t *size)

in CUDA 4.0.

Global CUDBGAPI_st::getGridDim32)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)

in CUDA 4.0.

Global CUDBGAPI_st::getGridStatus50)(uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)

in CUDA 5.5.

Global CUDBGAPI_st::getPhysicalRegister30)(uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)

in CUDA 3.1.

Global CUDBGAPI_st::getPhysicalRegister40)(uint32_t dev, uint32_t sm, uint32_t wp, uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)

in CUDA 4.1.

Global CUDBGNotifyNewEventCallback31

in CUDA 3.2.

Global CUDBGAPI_st::acknowledgeEvent30)(CUDBGEvent30 *event)

in CUDA 3.1.

Global CUDBGAPI_st::acknowledgeEvents42)(void)

in CUDA 5.0.

Global CUDBGAPI_st::getNextAsyncEvent50)(CUDBGEvent50 *event)

in CUDA 5.5.

Global CUDBGAPI_st::getNextEvent30)(CUDBGEvent30 *event)

in CUDA 3.1.

Global CUDBGAPI_st::getNextEvent32)(CUDBGEvent32 *event)

in CUDA 4.0

Global CUDBGAPI_st::getNextEvent42)(CUDBGEvent42 *event)

in CUDA 5.0

Global CUDBGAPI_st::getNextSyncEvent50)(CUDBGEvent50 *event)

in CUDA 5.5.

Global CUDBGAPI_st::setNotifyNewEventCallback31)
(CUDBGNotifyNewEventCallback31 callback, void *data)

in CUDA 3.2.

Global CUDBGAPI_st::setNotifyNewEventCallback40)
(CUDBGNotifyNewEventCallback40 callback)

in CUDA 4.1.