



NVIDIA CUDA DEBUGGER API

Reference Manual

Version 5.0

October 2012



Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, Tesla, and Quadro are trademarks or registered trademarks of NVIDIA Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2007-2012 NVIDIA Corporation. All rights reserved.



nVIDIA.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Contents

1	Introduction	1
1.1	Debugger API	1
1.2	ELF and DWARF	2
1.3	ABI Support	3
1.4	Exception Reporting	3
2	Module Index	5
2.1	Modules	5
3	Data Structure Index	7
3.1	Data Structures	7
4	File Index	9
4.1	File List	9
5	Module Documentation	11
5.1	Initialization	11
5.1.1	Detailed Description	11
5.1.2	Variable Documentation	11
5.1.2.1	finalize	11
5.1.2.2	initialize	11
5.2	Device Execution Control	13
5.2.1	Detailed Description	13
5.2.2	Variable Documentation	13
5.2.2.1	resumeDevice	13
5.2.2.2	singleStepWarp	13
5.2.2.3	singleStepWarp40	14
5.2.2.4	suspendDevice	14
5.3	Breakpoints	16
5.3.1	Detailed Description	16

5.3.2	Variable Documentation	16
5.3.2.1	setBreakpoint	16
5.3.2.2	setBreakpoint31	16
5.3.2.3	unsetBreakpoint	17
5.3.2.4	unsetBreakpoint31	17
5.4	Device State Inspection	18
5.4.1	Detailed Description	20
5.4.2	Variable Documentation	20
5.4.2.1	memcheckReadErrorAddress	20
5.4.2.2	readActiveLanes	21
5.4.2.3	readBlockIdx	21
5.4.2.4	readBlockIdx32	22
5.4.2.5	readBrokenWarps	22
5.4.2.6	readCallDepth	23
5.4.2.7	readCallDepth32	23
5.4.2.8	readCodeMemory	24
5.4.2.9	readConstMemory	24
5.4.2.10	readGlobalMemory	25
5.4.2.11	readGlobalMemory31	26
5.4.2.12	readGridId	26
5.4.2.13	readLaneException	27
5.4.2.14	readLaneStatus	27
5.4.2.15	readLocalMemory	28
5.4.2.16	readParamMemory	28
5.4.2.17	readPC	29
5.4.2.18	readPinnedMemory	30
5.4.2.19	readRegister	30
5.4.2.20	readReturnAddress	31
5.4.2.21	readReturnAddress32	31
5.4.2.22	readSharedMemory	32
5.4.2.23	readSyscallCallDepth	33
5.4.2.24	readTextureMemory	33
5.4.2.25	readTextureMemoryBindless	34
5.4.2.26	readThreadId	35
5.4.2.27	readValidLanes	35
5.4.2.28	readValidWarps	36
5.4.2.29	readVirtualPC	36

5.4.2.30	readVirtualReturnAddress	37
5.4.2.31	readVirtualReturnAddress32	37
5.4.2.32	writePinnedMemory	38
5.5	Device State Alteration	39
5.5.1	Detailed Description	39
5.5.2	Variable Documentation	39
5.5.2.1	writeGlobalMemory	39
5.5.2.2	writeGlobalMemory31	40
5.5.2.3	writeLocalMemory	40
5.5.2.4	writeParamMemory	41
5.5.2.5	writeRegister	42
5.5.2.6	writeSharedMemory	42
5.6	Grid Properties	44
5.6.1	Detailed Description	44
5.6.2	Variable Documentation	44
5.6.2.1	getBlockDim	44
5.6.2.2	getElfImage	45
5.6.2.3	getElfImage32	45
5.6.2.4	getGridAttribute	46
5.6.2.5	getGridAttributes	46
5.6.2.6	getGridDim	46
5.6.2.7	getGridDim32	47
5.6.2.8	getTID	47
5.7	Device Properties	48
5.7.1	Detailed Description	48
5.7.2	Variable Documentation	48
5.7.2.1	getDeviceType	48
5.7.2.2	getNumDevices	49
5.7.2.3	getNumLanes	49
5.7.2.4	getNumRegisters	49
5.7.2.5	getNumSMs	50
5.7.2.6	getNumWarps	50
5.7.2.7	getSmType	51
5.8	DWARF Utilities	52
5.8.1	Detailed Description	52
5.8.2	Variable Documentation	52
5.8.2.1	disassemble	52

5.8.2.2	getHostAddrFromDeviceAddr	53
5.8.2.3	getPhysicalRegister30	53
5.8.2.4	getPhysicalRegister40	53
5.8.2.5	isDeviceCodeAddress	54
5.8.2.6	lookupDeviceCodeSymbol	54
5.9	Events	55
5.9.1	Detailed Description	56
5.9.2	Enumeration Type Documentation	57
5.9.2.1	CUDBGEventKind	57
5.9.3	Variable Documentation	58
5.9.3.1	acknowledgeEvent30	58
5.9.3.2	acknowledgeEvents42	58
5.9.3.3	acknowledgeSyncEvents	58
5.9.3.4	getNextAsyncEvent	58
5.9.3.5	getNextEvent30	59
5.9.3.6	getNextEvent32	59
5.9.3.7	getNextEvent42	59
5.9.3.8	getNextSyncEvent	59
5.9.3.9	setNotifyNewEventCallback	60
5.9.3.10	setNotifyNewEventCallback31	60
5.9.3.11	setNotifyNewEventCallback40	60
6	Data Structure Documentation	61
6.1	cudbgGetAPI Struct Reference	61
6.1.1	Detailed Description	66
6.1.2	Field Documentation	67
6.1.2.1	clearAttachState	67
6.1.2.2	getGridStatus	67
6.1.2.3	requestCleanupOnDetach	67
6.2	CUDBGEvent Struct Reference	68
6.2.1	Detailed Description	68
6.3	CUDBGEvent30 Struct Reference	69
6.3.1	Detailed Description	69
6.4	CUDBGEvent30::CUDBGEvent30::cases30_st Union Reference	70
6.4.1	Detailed Description	70
6.5	CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::elfImageLoaded30_st Struct Reference	71

6.5.1 Detailed Description	71
6.6 CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelFinished30_st Struct Reference	72
6.6.1 Detailed Description	72
6.7 CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelReady30_st Struct Reference	73
6.7.1 Detailed Description	73
6.8 CUDBGEvent32 Struct Reference	74
6.8.1 Detailed Description	74
6.9 CUDBGEvent32::CUDBGEvent32::cases32_st Union Reference	75
6.9.1 Detailed Description	75
6.10 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextCreate32_st Struct Reference	76
6.10.1 Detailed Description	76
6.11 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextDestroy32_st Struct Reference	77
6.11.1 Detailed Description	77
6.12 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPop32_st Struct Reference	78
6.12.1 Detailed Description	78
6.13 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPush32_st Struct Reference	79
6.13.1 Detailed Description	79
6.14 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::elfImageLoaded32_st Struct Reference	80
6.14.1 Detailed Description	80
6.15 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelFinished32_st Struct Reference	81
6.15.1 Detailed Description	81
6.16 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelReady32_st Struct Reference	82
6.16.1 Detailed Description	82
6.17 CUDBGEvent42 Struct Reference	83
6.17.1 Detailed Description	83
6.18 CUDBGEvent42::CUDBGEvent42::cases42_st Union Reference	84
6.18.1 Detailed Description	84
6.19 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextCreate42_st Struct Reference	85
6.19.1 Detailed Description	85

6.20	CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextDestroy42_st Struct Reference	86
6.20.1	Detailed Description	86
6.21	CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPop42_st Struct Reference	87
6.21.1	Detailed Description	87
6.22	CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPush42_st Struct Reference	88
6.22.1	Detailed Description	88
6.23	CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::elfImageLoaded42_st Struct Reference	89
6.23.1	Detailed Description	89
6.24	CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelFinished42_st Struct Reference	90
6.24.1	Detailed Description	90
6.25	CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelReady42_st Struct Reference	91
6.25.1	Detailed Description	91
6.26	CUDBGEvent::CUDBGEvent::cases_st Union Reference	92
6.26.1	Detailed Description	92
6.27	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st Struct Reference	93
6.27.1	Detailed Description	93
6.28	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st Struct Reference	94
6.28.1	Detailed Description	94
6.29	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st Struct Reference	95
6.29.1	Detailed Description	95
6.30	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st Struct Reference	96
6.30.1	Detailed Description	96
6.31	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st Struct Reference	97
6.31.1	Detailed Description	97
6.32	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st Struct Reference	98
6.32.1	Detailed Description	98
6.33	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st Struct Reference	99
6.33.1	Detailed Description	99
6.34	CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st Struct Reference	100
6.34.1	Detailed Description	100
6.35	CUDBGEventCallbackData Struct Reference	101
6.35.1	Detailed Description	101
6.36	CUDBGEventCallbackData40 Struct Reference	102

6.36.1 Detailed Description	102
7 File Documentation	103
7.1 cudadebugger.h File Reference	103
7.1.1 Detailed Description	107
7.1.2 Enumeration Type Documentation	107
7.1.2.1 CUDBGAttribute	107
7.1.2.2 CUDBGException_t	107
7.1.2.3 CUDBGGridStatus	108
7.1.2.4 CUDBGKernelType	108
7.1.2.5 CUDBGRegClass	108
7.1.2.6 CUDBGResult	109
7.1.3 Function Documentation	110
7.1.3.1 cudbgGetAPIVersion	110

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 running, 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      (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).

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Initialization	11
Device Execution Control	13
Breakpoints	16
Device State Inspection	18
Device State Alteration	39
Grid Properties	44
Device Properties	48
DWARF Utilities	52
Events	55

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

cudbgGetAPI (The CUDA debugger API routines)	61
CUDBGEvent (Event information container)	68
CUDBGEvent30 (Event information container. Deprecated in 3.1)	69
CUDBGEvent30::CUDBGEvent30::cases30_st	70
CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::elfImageLoaded30_st	71
CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelFinished30_st	72
CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelReady30_st	73
CUDBGEvent32 (Event information container. Deprecated in 4.0)	74
CUDBGEvent32::CUDBGEvent32::cases32_st	75
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextCreate32_st	76
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextDestroy32_st	77
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPop32_st	78
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPush32_st	79
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::elfImageLoaded32_st	80
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelFinished32_st	81
CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelReady32_st	82
CUDBGEvent42 (Event information container. Deprecated in 4.2)	83
CUDBGEvent42::CUDBGEvent42::cases42_st	84
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextCreate42_st	85
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextDestroy42_st	86
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPop42_st	87
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPush42_st	88
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::elfImageLoaded42_st	89
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelFinished42_st	90
CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelReady42_st	91
CUDBGEvent::CUDBGEvent::cases_st	92
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st	93
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st	94
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st	95
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st	96
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st	97
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st	98
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st	99

CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st	100
CUDBGEventCallbackData (Event information passed to callback set with setNotifyNewEventCallback function)	101
CUDBGEventCallbackData40 (Event information passed to callback set with setNotifyNewEventCallback function. Deprecated in 4.1)	102

Chapter 4

File Index

4.1 File List

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

[cudadebugger.h](#) (Header file for the CUDA debugger API) 103

Chapter 5

Module Documentation

5.1 Initialization

Variables

- [CUDBGResult\(* CUDBGAPI_st::finalize\)](#)(void)
Finalize the API and free all memory.
- [CUDBGResult\(* CUDBGAPI_st::initialize\)](#)(void)
Initialize the API.

5.1.1 Detailed Description

5.1.2 Variable Documentation

5.1.2.1 `cudbgGetAPI::finalize` [inherited]

Finalize the API and free all memory.

Returns:

[CUDBG_SUCCESS](#),
[CUDBG_ERROR_UNINITIALIZED](#),
[CUDBG_ERROR_COMMUNICATION_FAILURE](#),
[CUDBG_ERROR_UNKNOWN](#)

See also:

[initialize](#)

5.1.2.2 `cudbgGetAPI::initialize` [inherited]

Initialize the API.

Returns:

CUDBG_SUCCESS,
CUDBG_ERROR_UNKNOWN

See also:

[finalize](#)

5.2 Device Execution Control

Variables

- [CUDBGResult\(* CUDBGAPI_st::resumeDevice \)\(uint32_t dev\)](#)
Resume a suspended CUDA device.
- [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.
- [CUDBGResult\(* CUDBGAPI_st::singleStepWarp40 \)\(uint32_t dev, uint32_t sm, uint32_t wp\)](#)
(DEPRECATED)Single step an individual warp on a suspended CUDA device. This function has been deprecated. Use [singleStepWarp\(\)](#) instead.
- [CUDBGResult\(* CUDBGAPI_st::suspendDevice \)\(uint32_t dev\)](#)
Suspends a running CUDA device.

5.2.1 Detailed Description

5.2.2 Variable Documentation

5.2.2.1 cudbgGetAPI::resumeDevice [inherited]

Resume a suspended CUDA device.

Parameters:

dev - device index

Returns:

CUDBG_SUCCESS,
CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_RUNNING_DEVICE,
CUDBG_ERROR_UNINITIALIZED

See also:

[suspendDevice](#)
[singleStepWarp](#)

5.2.2.2 cudbgGetAPI::singleStepWarp [inherited]

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

See also:

[resumeDevice](#)
[suspendDevice](#)

5.2.2.3 cudbgGetAPI::singleStepWarp40 [inherited]

(DEPRECATED)Single step an individual warp on a suspended CUDA device. This function has been deprecated. Use [singleStepWarp\(\)](#) instead.

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

See also:

[resumeDevice](#)
[suspendDevice](#)
[singleStepWarp](#)

5.2.2.4 cudbgGetAPI::suspendDevice [inherited]

Suspends a running CUDA device.

Parameters:

dev - device index

Returns:

CUDBG_SUCCESS,
CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_RUNNING_DEVICE,
CUDBG_ERROR_UNINITIALIZED

See also:

[resumeDevice](#)
[singleStepWarp](#)

5.3 Breakpoints

Variables

- [CUDBGResult\(* CUDBGAPI_st::setBreakpoint\)](#)(uint32_t dev, uint64_t addr)
Sets a breakpoint at the given instruction address for the given device.
- [CUDBGResult\(* CUDBGAPI_st::setBreakpoint31\)](#)(uint64_t addr)
Sets a breakpoint at the given instruction address. Deprecated in 3.2.
- [CUDBGResult\(* CUDBGAPI_st::unsetBreakpoint\)](#)(uint32_t dev, uint64_t addr)
Unsets a breakpoint at the given instruction address for the given device.
- [CUDBGResult\(* CUDBGAPI_st::unsetBreakpoint31\)](#)(uint64_t addr)
Unsets a breakpoint at the given instruction address. Deprecated in 3.2.

5.3.1 Detailed Description

5.3.2 Variable Documentation

5.3.2.1 `cudbgGetAPI::setBreakpoint` [inherited]

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

See also:

[unsetBreakpoint](#)

5.3.2.2 `cudbgGetAPI::setBreakpoint31` [inherited]

Sets a breakpoint at the given instruction address. Deprecated in 3.2.

Parameters:

addr - instruction address

Returns:

CUDBG_SUCCESS,
CUDBG_ERROR_UNINITIALIZED,
CUDBG_ERROR_INVALID_ADDRESS

See also:

[unsetBreakpoint31](#)

5.3.2.3 cudbgGetAPI::unsetBreakpoint [inherited]

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

See also:

[setBreakpoint](#)

5.3.2.4 cudbgGetAPI::unsetBreakpoint31 [inherited]

Unsets a breakpoint at the given instruction address. Deprecated in 3.2.

Parameters:

addr - instruction address

Returns:

CUDBG_SUCCESS,
CUDBG_ERROR_UNINITIALIZED

See also:

[setBreakpoint31](#)

5.4 Device State Inspection

Variables

- `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.
- `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.
- `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.
- `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. Deprecated in 4.0.
- `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.
- `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.
- `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. Deprecated in 4.0.
- `CUDBGResult(* CUDBGAPI_st::readCodeMemory)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)`
Reads content at address in the code memory segment.
- `CUDBGResult(* CUDBGAPI_st::readConstMemory)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)`
Reads content at address in the constant memory segment.
- `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+).
- `CUDBGResult(* CUDBGAPI_st::readGlobalMemory31)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)`
Reads content at address in the global memory segment. Deprecated in 3.2.
- `CUDBGResult(* CUDBGAPI_st::readGridId)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)`
Reads the CUDA grid index running on a valid warp.
- `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.

- `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`.
- `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.
- `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.
- `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.
- `CUDBGResult(* CUDBGAPI_st::readPinnedMemory)(uint64_t addr, void *buf, uint32_t sz)`
Reads content at pinned address in system memory.
- `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.
- `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.
- `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. Deprecated in 4.0.
- `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.
- `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.
- `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.
- `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.
- `CUDBGResult(* CUDBGAPI_st::readThreadId)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, CuDim3 *threadIdx)`
Reads the CUDA thread index running on valid lane.

- `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.
- `CUDBGResult(* CUDBGAPI_st::readValidWarps)(uint32_t dev, uint32_t sm, uint64_t *validWarpsMask)`
Reads the bitmask of valid warps on a given SM.
- `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.
- `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.
- `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. Deprecated in 4.0.
- `CUDBGResult(* CUDBGAPI_st::writePinnedMemory)(uint64_t addr, const void *buf, uint32_t sz)`
Writes content to pinned address in system memory.

5.4.1 Detailed Description

5.4.2 Variable Documentation

5.4.2.1 `cudbgGetAPI::memcheckReadErrorAddress` [inherited]

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`

5.4.2.2 `cudbgGetAPI::readActiveLanes` [inherited]

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

See also:

[readGridId](#)
[readBlockIdx](#)
[readThreadId](#)
[readBrokenWarps](#)
[readValidWarps](#)
[readValidLanes](#)

5.4.2.3 `cudbgGetAPI::readBlockIdx` [inherited]

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

See also:

[readGridId](#)
[readThreadId](#)
[readBrokenWarps](#)

[readValidWarps](#)
[readValidLanes](#)
[readActiveLanes](#)

5.4.2.4 `cudbgGetAPI::readBlockIdx32` [inherited]

Reads the two-dimensional CUDA block index running on a valid warp. Deprecated in 4.0.

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

See also:

[readGridId](#)
[readThreadId](#)
[readBrokenWarps](#)
[readValidWarps](#)
[readValidLanes](#)
[readActiveLanes](#)

5.4.2.5 `cudbgGetAPI::readBrokenWarps` [inherited]

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

See also:

[readGridId](#)
[readBlockIdx](#)
[readThreadIdx](#)
[readValidWarps](#)
[readValidLanes](#)
[readActiveLanes](#)

5.4.2.6 `cudbgGetAPI::readCallDepth` [inherited]

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

See also:

[readReturnAddress](#)
[readVirtualReturnAddress](#)

5.4.2.7 `cudbgGetAPI::readCallDepth32` [inherited]

Reads the call depth (number of calls) for a given warp. Deprecated in 4.0.

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

See also:

[readReturnAddress32](#)
[readVirtualReturnAddress32](#)

5.4.2.8 cudbgGetAPI::readCodeMemory [inherited]

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

See also:

[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.9 cudbgGetAPI::readConstMemory [inherited]

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

See also:

[readCodeMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.10 cudbgGetAPI::readGlobalMemory [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.11 `cudbgGetAPI::readGlobalMemory31` [inherited]

Reads content at address in the global memory segment. Deprecated in 3.2.

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.12 `cudbgGetAPI::readGridId` [inherited]

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

See also:

[readBlockIdx](#)

[readThreadId](#)
[readBrokenWarps](#)
[readValidWarps](#)
[readValidLanes](#)
[readActiveLanes](#)

5.4.2.13 `cudbgGetAPI::readLaneException` [inherited]

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

5.4.2.14 `cudbgGetAPI::readLaneStatus` [inherited]

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

5.4.2.15 cudbgGetAPI::readLocalMemory [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.16 cudbgGetAPI::readParamMemory [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.17 cudbgGetAPI::readPC [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readVirtualPC](#)

5.4.2.18 `cudbgGetAPI::readPinnedMemory` [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.19 `cudbgGetAPI::readRegister` [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readTextureMemory](#)
[readLocalMemory](#)
[readPC](#)

5.4.2.20 `cudbgGetAPI::readReturnAddress` [inherited]

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

See also:

[readCallDepth](#)
[readVirtualReturnAddress](#)

5.4.2.21 `cudbgGetAPI::readReturnAddress32` [inherited]

Reads the physical return address for a call level. Deprecated in 4.0.

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

See also:

[readCallDepth32](#)
[readVirtualReturnAddress32](#)

5.4.2.22 cudbgGetAPI::readSharedMemory [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readLocalMemory](#)
[readTextureMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.23 cudbgGetAPI::readSyscallCallDepth [inherited]

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

See also:

[readReturnAddress](#)
[readVirtualReturnAddress](#)

5.4.2.24 cudbgGetAPI::readTextureMemory [inherited]

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

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

On sm_30 and higher, use readTextureMemoryBindless instead.

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.25 cudbgGetAPI::readTextureMemoryBindless [inherited]

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

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

For sm_20 and lower, use readTextureMemory instead.

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.4.2.26 `cudbgGetAPI::readThreadId` [inherited]

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

See also:

[readGridId](#)
[readBlockIdx](#)
[readBrokenWarps](#)
[readValidWarps](#)
[readValidLanes](#)
[readActiveLanes](#)

5.4.2.27 `cudbgGetAPI::readValidLanes` [inherited]

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

See also:

[readGridId](#)
[readBlockIdx](#)
[readThreadIdx](#)
[readBrokenWarps](#)
[readValidWarps](#)
[readActiveLanes](#)

5.4.2.28 `cudbgGetAPI::readValidWarps` [inherited]

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

See also:

[readGridId](#)
[readBlockIdx](#)
[readThreadIdx](#)
[readBrokenWarps](#)
[readValidLanes](#)
[readActiveLanes](#)

5.4.2.29 `cudbgGetAPI::readVirtualPC` [inherited]

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

```

See also:

[readPC](#)

5.4.2.30 cudbgGetAPI::readVirtualReturnAddress [inherited]

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

```

See also:

[readCallDepth](#)
[readReturnAddress](#)

5.4.2.31 cudbgGetAPI::readVirtualReturnAddress32 [inherited]

Reads the virtual return address for a call level. Deprecated in 4.0.

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

See also:

[readCallDepth32](#)
[readReturnAddress32](#)

5.4.2.32 cudbgGetAPI::writePinnedMemory [inherited]

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

See also:

[readCodeMemory](#)
[readConstMemory](#)
[readGlobalMemory](#)
[readParamMemory](#)
[readSharedMemory](#)
[readLocalMemory](#)
[readRegister](#)
[readPC](#)

5.5 Device State Alteration

Variables

- `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+).
- `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. Deprecated in 3.2.
- `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.
- `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.
- `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.
- `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.

5.5.1 Detailed Description

5.5.2 Variable Documentation

5.5.2.1 `cudbgGetAPI::writeGlobalMemory` [inherited]

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
```

See also:

[writeParamMemory](#)
[writeSharedMemory](#)
[writeLocalMemory](#)
[writeRegister](#)

5.5.2.2 cudbgGetAPI::writeGlobalMemory31 [inherited]

Writes content to address in the global memory segment. Deprecated in 3.2.

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
```

See also:

[writeParamMemory](#)
[writeSharedMemory](#)
[writeLocalMemory](#)
[writeRegister](#)

5.5.2.3 cudbgGetAPI::writeLocalMemory [inherited]

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

See also:

[writeGlobalMemory](#)
[writeParamMemory](#)
[writeSharedMemory](#)
[writeRegister](#)

5.5.2.4 cudbgGetAPI::writeParamMemory [inherited]

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

See also:

[writeGlobalMemory](#)
[writeSharedMemory](#)
[writeLocalMemory](#)
[writeRegister](#)

5.5.2.5 cudbgGetAPI::writeRegister [inherited]

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

See also:

[writeGlobalMemory](#)
[writeParamMemory](#)
[writeSharedMemory](#)
[writeLocalMemory](#)

5.5.2.6 cudbgGetAPI::writeSharedMemory [inherited]

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

See also:

[writeGlobalMemory](#)
[writeParamMemory](#)
[writeLocalMemory](#)
[writeRegister](#)

5.6 Grid Properties

Variables

- `CUDBGResult(* CUDBGAPI_st::getBlockDim)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockDim)`
Get the number of threads in the given block.
- `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.
- `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. Deprecated in 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.
- `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.
- `CUDBGResult(* CUDBGAPI_st::getGridDim)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *gridDim)`
Get the number of blocks in the given grid.
- `CUDBGResult(* CUDBGAPI_st::getGridDim32)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)`
Get the number of blocks in the given grid. Deprecated in 4.0.
- `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.

5.6.1 Detailed Description

5.6.2 Variable Documentation

5.6.2.1 `cudbgGetAPI::getBlockDim` [inherited]

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

See also:

[getGridDim](#)

5.6.2.2 cudbgGetAPI::getElfImage [inherited]

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

5.6.2.3 cudbgGetAPI::getElfImage32 [inherited]

Get the relocated or non-relocated ELF image and size for the grid on the given device. Deprecated in 4.0.

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

5.6.2.4 cudbgGetAPI::getGridAttribute [inherited]

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

5.6.2.5 cudbgGetAPI::getGridAttributes [inherited]

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

5.6.2.6 cudbgGetAPI::getGridDim [inherited]

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

See also:

[getBlockDim](#)

5.6.2.7 cudbgGetAPI::getGridDim32 [inherited]

Get the number of blocks in the given grid. Deprecated in 4.0.

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

See also:

[getBlockDim](#)

5.6.2.8 cudbgGetAPI::getTID [inherited]

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

5.7 Device Properties

Variables

- `CUDBGResult(* CUDBGAPI_st::getDeviceType)(uint32_t dev, char *buf, uint32_t sz)`
Get the string description of the device.
- `CUDBGResult(* CUDBGAPI_st::getNumDevices)(uint32_t *numDev)`
Get the number of installed CUDA devices.
- `CUDBGResult(* CUDBGAPI_st::getNumLanes)(uint32_t dev, uint32_t *numLanes)`
Get the number of lanes per warp on the device.
- `CUDBGResult(* CUDBGAPI_st::getNumRegisters)(uint32_t dev, uint32_t *numRegs)`
Get the number of registers per lane on the device.
- `CUDBGResult(* CUDBGAPI_st::getNumSMs)(uint32_t dev, uint32_t *numSMs)`
Get the total number of SMs on the device.
- `CUDBGResult(* CUDBGAPI_st::getNumWarps)(uint32_t dev, uint32_t *numWarps)`
Get the number of warps per SM on the device.
- `CUDBGResult(* CUDBGAPI_st::getSmType)(uint32_t dev, char *buf, uint32_t sz)`
Get the SM type of the device.

5.7.1 Detailed Description

5.7.2 Variable Documentation

5.7.2.1 `cudbgGetAPI::getDeviceType` [inherited]

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

See also:

`getSMType`

5.7.2.2 `cudbgGetAPI::getNumDevices` [inherited]

Get the number of installed CUDA devices.

Parameters:

numDev - the returned number of devices

Returns:

CUDBG_SUCCESS,
CUDBG_ERROR_INVALID_ARGS,
CUDBG_ERROR_UNINITIALIZED

See also:

[getNumSMs](#)
[getNumWarps](#)
[getNumLanes](#)
[getNumRegisters](#)

5.7.2.3 `cudbgGetAPI::getNumLanes` [inherited]

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

See also:

[getNumDevices](#)
[getNumSMs](#)
[getNumWarps](#)
[getNumRegisters](#)

5.7.2.4 `cudbgGetAPI::getNumRegisters` [inherited]

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

See also:

[getNumDevices](#)
[getNumSMs](#)
[getNumWarps](#)
[getNumLanes](#)

5.7.2.5 cudbgGetAPI::getNumSMs [inherited]

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

See also:

[getNumDevices](#)
[getNumWarps](#)
[getNumLanes](#)
[getNumRegisters](#)

5.7.2.6 cudbgGetAPI::getNumWarps [inherited]

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

See also:

[getNumDevices](#)
[getNumSMs](#)
[getNumLanes](#)
[getNumRegisters](#)

5.7.2.7 cudbgGetAPI::getSmType [inherited]

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

See also:

[getDeviceType](#)

5.8 DWARF Utilities

Variables

- `CUDBGResult(* CUDBGAPI_st::disassemble)(uint32_t dev, uint64_t addr, uint32_t *instSize, char *buf, uint32_t sz)`
Disassemble instruction at instruction address.
- `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.
- `CUDBGResult(* CUDBGAPI_st::getPhysicalRegister30)(uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)`
(DEPRECATED) Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC. The function has been deprecated. use getWarpPhysicalRegister instead.
- `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.
- `CUDBGResult(* CUDBGAPI_st::isDeviceCodeAddress)(uintptr_t addr, bool *isDeviceAddress)`
Determines whether a virtual address resides within device code.
- `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.

5.8.1 Detailed Description

5.8.2 Variable Documentation

5.8.2.1 cudbgGetAPI::disassemble [inherited]

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

5.8.2.2 cudbgGetAPI::getHostAddrFromDeviceAddr [inherited]

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

See also:

[readGlobalMemory](#)
[writeGlobalMemory](#)

5.8.2.3 cudbgGetAPI::getPhysicalRegister30 [inherited]

(DEPRECATED) Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC. The function has been deprecated. use getWarpPhysicalRegister instead.

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_UNKNOWN_FUNCTION,
CUDBG_ERROR_UNKNOWN

5.8.2.4 cudbgGetAPI::getPhysicalRegister40 [inherited]

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

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 the physical register with the highest physical register index will contain the lowest bits.

Parameters:

dev - device index
sm - SM index
wp - warp index
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_UNKNOWN_FUNCTION,
 CUDBG_ERROR_UNKNOWN

5.8.2.5 cudbgGetAPI::isDeviceCodeAddress [inherited]

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

5.8.2.6 cudbgGetAPI::lookupDeviceCodeSymbol [inherited]

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

5.9 Events

Data Structures

- struct [CUDBGEvent](#)
Event information container.
- struct [CUDBGEvent30](#)
Event information container. Deprecated in 3.1.
- struct [CUDBGEvent32](#)
Event information container. Deprecated in 4.0.
- struct [CUDBGEvent42](#)
Event information container. Deprecated in 4.2.
- struct [CUDBGEventCallbackData](#)
Event information passed to callback set with `setNotifyNewEventCallback` function.
- struct [CUDBGEventCallbackData40](#)
Event information passed to callback set with `setNotifyNewEventCallback` function. Deprecated in 4.1.

Typedefs

- typedef void(* [CUDBGNotifyNewEventCallback](#))(CUDBGEventCallbackData *data)
function type of the function called to notify debugger of the presence of a new event in the event queue.
- typedef void(* [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 3.2.

Enumerations

- enum [CUDBGEventKind](#) {
[CUDBG_EVENT_INVALID](#),
[CUDBG_EVENT_ELF_IMAGE_LOADED](#),
[CUDBG_EVENT_KERNEL_READY](#),
[CUDBG_EVENT_KERNEL_FINISHED](#),
[CUDBG_EVENT_INTERNAL_ERROR](#),
[CUDBG_EVENT_CTX_PUSH](#),
[CUDBG_EVENT_CTX_POP](#),
[CUDBG_EVENT_CTX_CREATE](#),
[CUDBG_EVENT_CTX_DESTROY](#),
[CUDBG_EVENT_TIMEOUT](#),
[CUDBG_EVENT_ATTACH_COMPLETE](#) }
CUDA Kernel Events.

Variables

- `CUDBGResult(* CUDBGAPI_st::acknowledgeEvent30)(CUDBGEvent30 *event)`
Inform the debugger API that the event has been processed. Deprecated in 3.1.
- `CUDBGResult(* CUDBGAPI_st::acknowledgeEvents42)(void)`
Inform the debugger API that synchronous events have been processed. Deprecated in 5.0.
- `CUDBGResult(* CUDBGAPI_st::acknowledgeSyncEvents)(void)`
Inform the debugger API that synchronous events have been processed.
- `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.
- `CUDBGResult(* CUDBGAPI_st::getNextEvent30)(CUDBGEvent30 *event)`
Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 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. Deprecated in 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. Deprecated in 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.
- `CUDBGResult(* CUDBGAPI_st::setNotifyNewEventCallback)(CUDBGNotifyNewEventCallback call-back)`
Provides the API with the function to call to notify the debugger of a new application or device event.
- `CUDBGResult(* CUDBGAPI_st::setNotifyNewEventCallback31)(CUDBGNotifyNewEventCallback31 call-back, void *data)`
Provides the API with the function to call to notify the debugger of a new application or device event. Deprecated in 3.2.
- `CUDBGResult(* CUDBGAPI_st::setNotifyNewEventCallback40)(CUDBGNotifyNewEventCallback40 call-back)`
Provides the API with the function to call to notify the debugger of a new application or device event. Deprecated in 4.1.

5.9.1 Detailed Description

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](#).

5.9.2 Enumeration Type Documentation

5.9.2.1 enum CUDBGEventKind

CUDA Kernel Events.

Enumerator:

CUDBG_EVENT_INVALID Invalid event.

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

CUDBG_EVENT_KERNEL_READY A CUDA kernel is about to be launched.

CUDBG_EVENT_KERNEL_FINISHED A CUDA kernel has terminated.

CUDBG_EVENT_INTERNAL_ERROR An internal error occur. The debugging framework may be unstable.

CUDBG_EVENT_CTX_PUSH A CUDA context was pushed.

CUDBG_EVENT_CTX_POP A CUDA CTX was popped.

CUDBG_EVENT_CTX_CREATE A CUDA CTX was created.

CUDBG_EVENT_CTX_DESTROY A CUDA context was destroyed.

CUDBG_EVENT_TIMEOUT An timeout event is sent at regular interval. This event can safely ge ignored.

CUDBG_EVENT_ATTACH_COMPLETE The attach process has completed and debugging of device code may start.

5.9.3 Variable Documentation

5.9.3.1 cudbgGetAPI::acknowledgeEvent30 [inherited]

Inform the debugger API that the event has been processed. Deprecated in 3.1.

Parameters:

event - pointer to the event that has been processed

Returns:

CUDBG_SUCCESS

5.9.3.2 cudbgGetAPI::acknowledgeEvents42 [inherited]

Inform the debugger API that synchronous events have been processed. Deprecated in 5.0.

Returns:

CUDBG_SUCCESS

5.9.3.3 cudbgGetAPI::acknowledgeSyncEvents [inherited]

Inform the debugger API that synchronous events have been processed.

Returns:

CUDBG_SUCCESS

5.9.3.4 cudbgGetAPI::getNextAsyncEvent [inherited]

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

5.9.3.5 cudbgGetAPI::getNextEvent30 [inherited]

Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 3.1.

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

5.9.3.6 cudbgGetAPI::getNextEvent32 [inherited]

Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 4.0.

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

5.9.3.7 cudbgGetAPI::getNextEvent42 [inherited]

Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 5.0.

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

5.9.3.8 cudbgGetAPI::getNextSyncEvent [inherited]

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

5.9.3.9 cudbgGetAPI::setNotifyNewEventCallback [inherited]

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

5.9.3.10 cudbgGetAPI::setNotifyNewEventCallback31 [inherited]

Provides the API with the function to call to notify the debugger of a new application or device event. Deprecated in 3.2.

Parameters:

callback - the callback function

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

Returns:

CUDBG_SUCCESS

5.9.3.11 cudbgGetAPI::setNotifyNewEventCallback40 [inherited]

Provides the API with the function to call to notify the debugger of a new application or device event. Deprecated in 4.1.

Parameters:

callback - the callback function

Returns:

CUDBG_SUCCESS

Chapter 6

Data Structure Documentation

6.1 cudbgGetAPI Struct Reference

The CUDA debugger API routines.

Data Fields

- [CUDBGResult\(* acknowledgeEvent30\)](#)(CUDBGEvent30 *event)
Inform the debugger API that the event has been processed. Deprecated in 3.1.
- [CUDBGResult\(* acknowledgeEvents42\)](#)(void)
Inform the debugger API that synchronous events have been processed. Deprecated in 5.0.
- [CUDBGResult\(* acknowledgeSyncEvents\)](#)(void)
Inform the debugger API that synchronous events have been processed.
- [CUDBGResult\(* clearAttachState\)](#)(void)
Clear attach-specific state prior to detach.
- [CUDBGResult\(* disassemble\)](#)(uint32_t dev, uint64_t addr, uint32_t *instSize, char *buf, uint32_t sz)
Disassemble instruction at instruction address.
- [CUDBGResult\(* finalize\)](#)(void)
Finalize the API and free all memory.
- [CUDBGResult\(* getBlockDim\)](#)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockDim)
Get the number of threads in the given block.
- [CUDBGResult\(* getDeviceType\)](#)(uint32_t dev, char *buf, uint32_t sz)
Get the string description of the device.
- [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.

- **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. Deprecated in 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.
- **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.
- **CUDBGResult**(* [getGridDim](#))(uint32_t dev, uint32_t sm, uint32_t wp, [CuDim3](#) *gridDim)
Get the number of blocks in the given grid.
- **CUDBGResult**(* [getGridDim32](#))(uint32_t dev, uint32_t sm, uint32_t wp, [CuDim2](#) *gridDim)
Get the number of blocks in the given grid. Deprecated in 4.0.
- **CUDBGResult**(* [getGridStatus](#))(uint32_t dev, uint32_t gridId, [CUDBGGridStatus](#) *status)
Check whether the grid corresponding to the given gridId is still present on the device.
- **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.
- **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.
- **CUDBGResult**(* [getNextEvent30](#))(CUDBGEvent30 *event)
Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 3.1.
- **CUDBGResult**(* [getNextEvent32](#))(CUDBGEvent32 *event)
Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 4.0.
- **CUDBGResult**(* [getNextEvent42](#))(CUDBGEvent42 *event)
Copies the next available event in the event queue into 'event' and removes it from the queue. Deprecated in 5.0.
- **CUDBGResult**(* [getNextSyncEvent](#))(CUDBGEvent *event)
Copies the next available event in the synchronous event queue into 'event' and removes it from the queue.
- **CUDBGResult**(* [getNumDevices](#))(uint32_t *numDev)
Get the number of installed CUDA devices.
- **CUDBGResult**(* [getNumLanes](#))(uint32_t dev, uint32_t *numLanes)
Get the number of lanes per warp on the device.
- **CUDBGResult**(* [getNumRegisters](#))(uint32_t dev, uint32_t *numRegs)
Get the number of registers per lane on the device.
- **CUDBGResult**(* [getNumSMs](#))(uint32_t dev, uint32_t *numSMs)

Get the total number of SMs on the device.

- **CUDBGResult(* [getNumWarps](#))**(uint32_t dev, uint32_t *numWarps)
Get the number of warps per SM on the device.
- **CUDBGResult(* [getPhysicalRegister30](#))**(uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, **CUDBGRegClass** *regClass)
(DEPRECATED) Get the physical register number(s) assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC. The function has been deprecated. use [getWarpPhysicalRegister](#) instead.
- **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.
- **CUDBGResult(* [getSmType](#))**(uint32_t dev, char *buf, uint32_t sz)
Get the SM type of the device.
- **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.
- **CUDBGResult(* [initialize](#))**(void)
Initialize the API.
- **CUDBGResult(* [isDeviceCodeAddress](#))**(uintptr_t addr, bool *isDeviceAddress)
Determines whether a virtual address resides within device code.
- **CUDBGResult(* [lookupDeviceCodeSymbol](#))**(char *symName, bool *symFound, uintptr_t *symAddr)
Determines whether a symbol represents a function in device code and returns its virtual address.
- **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.
- **CUDBGResult(* [readActiveLanes](#))**(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *activeLanesMask)
Reads the bitmask of active lanes on a valid warp.
- **CUDBGResult(* [readBlockIdx](#))**(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockIdx)
Reads the CUDA block index running on a valid warp.
- **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. Deprecated in 4.0.
- **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.
- **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.
- **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. Deprecated in 4.0.

- [CUDBGResult\(* readCodeMemory\)](#)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the code memory segment.
- [CUDBGResult\(* readConstMemory\)](#)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the constant memory segment.
- [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+).
- [CUDBGResult\(* readGlobalMemory31\)](#)(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the global memory segment. Deprecated in 3.2.
- [CUDBGResult\(* readGridId\)](#)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)
Reads the CUDA grid index running on a valid warp.
- [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.
- [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.
- [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.
- [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.
- [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.
- [CUDBGResult\(* readPinnedMemory\)](#)(uint64_t addr, void *buf, uint32_t sz)
Reads content at pinned address in system memory.
- [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.
- [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.
- [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. Deprecated in 4.0.
- [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.

- **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.
- **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.
- **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.
- **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.
- **CUDBGResult(* readValidLanes)**(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *validLanesMask)
Reads the bitmask of valid lanes on a given warp.
- **CUDBGResult(* readValidWarps)**(uint32_t dev, uint32_t sm, uint64_t *validWarpsMask)
Reads the bitmask of valid warps on a given SM.
- **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.
- **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.
- **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. Deprecated in 4.0.
- **CUDBGResult(* requestCleanupOnDetach)**(void)
Request for cleanup of driver state when detaching.
- **CUDBGResult(* resumeDevice)**(uint32_t dev)
Resume a suspended CUDA device.
- **CUDBGResult(* setBreakpoint)**(uint32_t dev, uint64_t addr)
Sets a breakpoint at the given instruction address for the given device.
- **CUDBGResult(* setBreakpoint31)**(uint64_t addr)
Sets a breakpoint at the given instruction address. Deprecated in 3.2.
- **CUDBGResult(* setNotifyNewEventCallback)**(CUDBGNotifyNewEventCallback callback)
Provides the API with the function to call to notify the debugger of a new application or device event.
- **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. Deprecated in 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. Deprecated in 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.
- [CUDBGResult\(* singleStepWarp40\)](#)(uint32_t dev, uint32_t sm, uint32_t wp)
(DEPRECATED)Single step an individual warp on a suspended CUDA device. This function has been deprecated. Use [singleStepWarp\(\)](#) instead.
- [CUDBGResult\(* suspendDevice\)](#)(uint32_t dev)
Suspends a running CUDA device.
- [CUDBGResult\(* unsetBreakpoint\)](#)(uint32_t dev, uint64_t addr)
Unsets a breakpoint at the given instruction address for the given device.
- [CUDBGResult\(* unsetBreakpoint31\)](#)(uint64_t addr)
Unsets a breakpoint at the given instruction address. Deprecated in 3.2.
- [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+).
- [CUDBGResult\(* writeGlobalMemory31\)](#)(uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)
Writes content to address in the global memory segment. Deprecated in 3.2.
- [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.
- [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.
- [CUDBGResult\(* writePinnedMemory\)](#)(uint64_t addr, const void *buf, uint32_t sz)
Writes content to pinned address in system memory.
- [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.
- [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.

6.1.1 Detailed Description

The CUDA debugger API routines.

6.1.2 Field Documentation

6.1.2.1 cudbgGetAPI::clearAttachState

Clear attach-specific state prior to detach.

Returns:

CUDBG_SUCCESS

6.1.2.2 cudbgGetAPI::getGridStatus

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

Parameters:

devId - device index

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

6.1.2.3 cudbgGetAPI::requestCleanupOnDetach

Request for cleanup of driver state when detaching.

Returns:

CUDBG_SUCCESS

6.2 CUDBGEvent Struct Reference

Event information container.

Data Structures

- union [cases_st](#)

Data Fields

- union [CUDBGEvent::cases_st](#) [cases](#)
Information for each type of event.
- [CUDBGEventKind](#) [kind](#)
Event type.

6.2.1 Detailed Description

Event information container.

6.3 CUDBGEvent30 Struct Reference

Event information container. Deprecated in 3.1.

Data Structures

- union [cases30_st](#)

Data Fields

- union [CUDBGEvent30::cases30_st](#) [cases](#)
Information for each type of event.
- [CUDBGEventKind](#) [kind](#)
Event type.

6.3.1 Detailed Description

Event information container. Deprecated in 3.1.

6.4 CUDBGEvent30::CUDBGEvent30::cases30_st Union Reference

Data Structures

- struct [elfImageLoaded30_st](#)
- struct [kernelFinished30_st](#)
- struct [kernelReady30_st](#)

Data Fields

- struct [CUDBGEvent30::cases30_st::elfImageLoaded30_st](#) [elfImageLoaded](#)
Information about the loaded ELF image.
- struct [CUDBGEvent30::cases30_st::kernelFinished30_st](#) [kernelFinished](#)
Information about the kernel that just terminated.
- struct [CUDBGEvent30::cases30_st::kernelReady30_st](#) [kernelReady](#)
Information about the kernel ready to be launched.

6.4.1 Detailed Description

6.5 CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::elfImageLoaded30_st Struct Reference

Data Fields

- char * [nonRelocatedElfImage](#)
pointer to the non-relocated ELF image for a CUDA source module.
- char * [relocatedElfImage](#)
pointer to the relocated ELF image for a CUDA source module.
- uint32_t [size](#)
size of the ELF image (32-bit).

6.5.1 Detailed Description

6.6 CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelFinished30_st Struct Reference

Data Fields

- uint32_t [dev](#)
device index of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).

6.6.1 Detailed Description

6.7 CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelReady30_st Struct Reference

Data Fields

- uint32_t [dev](#)
device index of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).

6.7.1 Detailed Description

6.8 CUDBGEvent32 Struct Reference

Event information container. Deprecated in 4.0.

Data Structures

- union [cases32_st](#)

Data Fields

- union [CUDBGEvent32::cases32_st](#) [cases](#)
Information for each type of event.
- [CUDBGEventKind](#) [kind](#)
Event type.

6.8.1 Detailed Description

Event information container. Deprecated in 4.0.

6.9 CUDBGEvent32::CUDBGEvent32::cases32_st Union Reference

Data Structures

- struct [contextCreate32_st](#)
- struct [contextDestroy32_st](#)
- struct [contextPop32_st](#)
- struct [contextPush32_st](#)
- struct [elfImageLoaded32_st](#)
- struct [kernelFinished32_st](#)
- struct [kernelReady32_st](#)

Data Fields

- struct [CUDBGEvent32::cases32_st::contextCreate32_st](#) contextCreate
Information about the context being created.
- struct [CUDBGEvent32::cases32_st::contextDestroy32_st](#) contextDestroy
Information about the context being destroyed.
- struct [CUDBGEvent32::cases32_st::contextPop32_st](#) contextPop
Information about the context being popped.
- struct [CUDBGEvent32::cases32_st::contextPush32_st](#) contextPush
Information about the context being pushed.
- struct [CUDBGEvent32::cases32_st::elfImageLoaded32_st](#) elfImageLoaded
Information about the loaded ELF image.
- struct [CUDBGEvent32::cases32_st::kernelFinished32_st](#) kernelFinished
Information about the kernel that just terminated.
- struct [CUDBGEvent32::cases32_st::kernelReady32_st](#) kernelReady
Information about the kernel ready to be launched.

6.9.1 Detailed Description

6.10 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextCreate32_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being created.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.10.1 Detailed Description

6.11 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextDestroy32_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being destroyed.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.11.1 Detailed Description

6.12 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPop32_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being popped.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.12.1 Detailed Description

6.13 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPush32_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being pushed.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.13.1 Detailed Description

6.14 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::elfImageLoaded32_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [module](#)
module of the kernel.
- char * [nonRelocatedElfImage](#)
pointer to the non-relocated ELF image for a CUDA source module.
- char * [relocatedElfImage](#)
pointer to the relocated ELF image for a CUDA source module.
- uint32_t [size](#)
size of the ELF image (32-bit).

6.14.1 Detailed Description

6.15 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelFinished32_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [function](#)
function of the kernel.
- uint64_t [functionEntry](#)
entry PC of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint64_t [module](#)
module of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).

6.15.1 Detailed Description

6.16 CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelReady32_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [function](#)
function of the kernel.
- uint64_t [functionEntry](#)
entry PC of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint64_t [module](#)
module of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).

6.16.1 Detailed Description

6.17 CUDBGEvent42 Struct Reference

Event information container. Deprecated in 4.2.

Data Structures

- union [cases42_st](#)

Data Fields

- union [CUDBGEvent42::cases42_st](#) [cases](#)
Information for each type of event.
- [CUDBGEventKind](#) [kind](#)
Event type.

6.17.1 Detailed Description

Event information container. Deprecated in 4.2.

6.18 CUDBGEvent42::CUDBGEvent42::cases42_st Union Reference

Data Structures

- struct [contextCreate42_st](#)
- struct [contextDestroy42_st](#)
- struct [contextPop42_st](#)
- struct [contextPush42_st](#)
- struct [elfImageLoaded42_st](#)
- struct [kernelFinished42_st](#)
- struct [kernelReady42_st](#)

Data Fields

- struct [CUDBGEvent42::cases42_st::contextCreate42_st](#) contextCreate
Information about the context being created.
- struct [CUDBGEvent42::cases42_st::contextDestroy42_st](#) contextDestroy
Information about the context being destroyed.
- struct [CUDBGEvent42::cases42_st::contextPop42_st](#) contextPop
Information about the context being popped.
- struct [CUDBGEvent42::cases42_st::contextPush42_st](#) contextPush
Information about the context being pushed.
- struct [CUDBGEvent42::cases42_st::elfImageLoaded42_st](#) elfImageLoaded
Information about the loaded ELF image.
- struct [CUDBGEvent42::cases42_st::kernelFinished42_st](#) kernelFinished
Information about the kernel that just terminated.
- struct [CUDBGEvent42::cases42_st::kernelReady42_st](#) kernelReady
Information about the kernel ready to be launched.

6.18.1 Detailed Description

6.19 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextCreate42_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being created.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.19.1 Detailed Description

6.20 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextDestroy42_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being destroyed.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.20.1 Detailed Description

6.21 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPop42_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being popped.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.21.1 Detailed Description

6.22 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPush42_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being pushed.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.22.1 Detailed Description

6.23 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::elfImageLoaded42_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [module](#)
module of the kernel.
- char * [nonRelocatedElfImage](#)
pointer to the non-relocated ELF image for a CUDA source module.
- char * [relocatedElfImage](#)
pointer to the relocated ELF image for a CUDA source module.
- uint64_t [size](#)
size of the ELF image (64-bit).
- uint32_t [size32](#)
size of the ELF image (32-bit). Deprecated in 4.0.

6.23.1 Detailed Description

6.24 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelFinished42_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [function](#)
function of the kernel.
- uint64_t [functionEntry](#)
entry PC of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint64_t [module](#)
module of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).

6.24.1 Detailed Description

6.25 CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelReady42_st Struct Reference

Data Fields

- CuDim3 [blockDim](#)
block dimensions of the kernel.
- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [function](#)
function of the kernel.
- uint64_t [functionEntry](#)
entry PC of the kernel.
- CuDim3 [gridDim](#)
grid dimensions of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint64_t [module](#)
module of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).
- [CUDBGKernelType](#) [type](#)
the type of the kernel: system or application.

6.25.1 Detailed Description

6.26 CUDBGEvent::CUDBGEvent::cases_st Union Reference

Data Structures

- struct [contextCreate_st](#)
- struct [contextDestroy_st](#)
- struct [contextPop_st](#)
- struct [contextPush_st](#)
- struct [elfImageLoaded_st](#)
- struct [internalError_st](#)
- struct [kernelFinished_st](#)
- struct [kernelReady_st](#)

Data Fields

- struct [CUDBGEvent::cases_st::contextCreate_st](#) contextCreate
Information about the context being created.
- struct [CUDBGEvent::cases_st::contextDestroy_st](#) contextDestroy
Information about the context being destroyed.
- struct [CUDBGEvent::cases_st::contextPop_st](#) contextPop
Information about the context being popped.
- struct [CUDBGEvent::cases_st::contextPush_st](#) contextPush
Information about the context being pushed.
- struct [CUDBGEvent::cases_st::elfImageLoaded_st](#) elfImageLoaded
Information about the loaded ELF image.
- struct [CUDBGEvent::cases_st::internalError_st](#) internalError
Information about internal erros.
- struct [CUDBGEvent::cases_st::kernelFinished_st](#) kernelFinished
Information about the kernel that just terminated.
- struct [CUDBGEvent::cases_st::kernelReady_st](#) kernelReady
Information about the kernel ready to be launched.

6.26.1 Detailed Description

6.27 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being created.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.27.1 Detailed Description

6.28 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being destroyed.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.28.1 Detailed Description

6.29 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being popped.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.29.1 Detailed Description

6.30 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st Struct Reference

Data Fields

- uint64_t [context](#)
the context being pushed.
- uint32_t [dev](#)
device index of the context.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the context (Linux only).

6.30.1 Detailed Description

6.31 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [module](#)
module of the kernel.
- char * [nonRelocatedElfImage](#)
pointer to the non-relocated ELF image for a CUDA source module.
- char * [relocatedElfImage](#)
pointer to the relocated ELF image for a CUDA source module.
- uint64_t [size](#)
size of the ELF image (64-bit).
- uint32_t [size32](#)
size of the ELF image (32-bit). Deprecated in 4.0.

6.31.1 Detailed Description

6.32 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st Struct Reference

Data Fields

- [CUDBGResult errorType](#)
Type of the internal error.

6.32.1 Detailed Description

6.33 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st Struct Reference

Data Fields

- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [function](#)
function of the kernel.
- uint64_t [functionEntry](#)
entry PC of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint64_t [module](#)
module of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).

6.33.1 Detailed Description

6.34 CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st Struct Reference

Data Fields

- CuDim3 [blockDim](#)
block dimensions of the kernel.
- uint64_t [context](#)
context of the kernel.
- uint32_t [dev](#)
device index of the kernel.
- uint64_t [function](#)
function of the kernel.
- uint64_t [functionEntry](#)
entry PC of the kernel.
- CuDim3 [gridDim](#)
grid dimensions of the kernel.
- uint32_t [gridId](#)
grid index of the kernel.
- uint64_t [module](#)
module of the kernel.
- uint32_t [tid](#)
host thread id (or LWP id) of the thread hosting the kernel (Linux only).
- [CUDBGKernelType](#) [type](#)
the type of the kernel: system or application.

6.34.1 Detailed Description

6.35 CUDBGEventCallbackData Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function.

Data Fields

- uint32_t [tid](#)

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

- uint32_t [timeout](#)

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

6.35.1 Detailed Description

Event information passed to callback set with setNotifyNewEventCallback function.

6.36 CUDBGEventCallbackData40 Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function. Deprecated in 4.1.

Data Fields

- uint32_t [tid](#)

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

6.36.1 Detailed Description

Event information passed to callback set with setNotifyNewEventCallback function. Deprecated in 4.1.

Chapter 7

File Documentation

7.1 cudadebugger.h File Reference

Header file for the CUDA debugger API.

Data Structures

- struct [cudbgGetAPI](#)
The CUDA debugger API routines.
- struct [CUDBGEvent](#)
Event information container.
- struct [CUDBGEvent30](#)
Event information container. Deprecated in 3.1.
- union [CUDBGEvent30::CUDBGEvent30::cases30_st](#)
- struct [CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::elfImageLoaded30_st](#)
- struct [CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelFinished30_st](#)
- struct [CUDBGEvent30::CUDBGEvent30::cases30_st::CUDBGEvent30::cases30_st::kernelReady30_st](#)
- struct [CUDBGEvent32](#)
Event information container. Deprecated in 4.0.
- union [CUDBGEvent32::CUDBGEvent32::cases32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextCreate32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextDestroy32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPop32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::contextPush32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::elfImageLoaded32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelFinished32_st](#)
- struct [CUDBGEvent32::CUDBGEvent32::cases32_st::CUDBGEvent32::cases32_st::kernelReady32_st](#)
- struct [CUDBGEvent42](#)
Event information container. Deprecated in 4.2.
- union [CUDBGEvent42::CUDBGEvent42::cases42_st](#)

- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextCreate42_st](#)
- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextDestroy42_st](#)
- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPop42_st](#)
- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::contextPush42_st](#)
- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::elfImageLoaded42_st](#)
- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelFinished42_st](#)
- struct [CUDBGEvent42::CUDBGEvent42::cases42_st::CUDBGEvent42::cases42_st::kernelReady42_st](#)
- union [CUDBGEvent::CUDBGEvent::cases_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st](#)
- struct [CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st](#)
- struct [CUDBGEventCallbackData](#)

Event information passed to callback set with [setNotifyNewEventCallback](#) function.

- struct [CUDBGEventCallbackData40](#)

Event information passed to callback set with [setNotifyNewEventCallback](#) function. Deprecated in 4.1.

Typedefs

- typedef void(* [CUDBGNotifyNewEventCallback](#))(CUDBGEventCallbackData *data)
function type of the function called to notify debugger of the presence of a new event in the event queue.
- typedef void(* [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 3.2.

Enumerations

- enum [CUDBGAttribute](#) {
 [CUDBG_ATTR_GRID_LAUNCH_BLOCKING](#),
 [CUDBG_ATTR_GRID_TID](#) }
Query attribute.
- enum [CUDBGEventKind](#) {
 [CUDBG_EVENT_INVALID](#),
 [CUDBG_EVENT_ELF_IMAGE_LOADED](#),
 [CUDBG_EVENT_KERNEL_READY](#),
 [CUDBG_EVENT_KERNEL_FINISHED](#),
 [CUDBG_EVENT_INTERNAL_ERROR](#),
 [CUDBG_EVENT_CTX_PUSH](#),
 [CUDBG_EVENT_CTX_POP](#),

```

CUDBG_EVENT_CTX_CREATE,
CUDBG_EVENT_CTX_DESTROY,
CUDBG_EVENT_TIMEOUT,
CUDBG_EVENT_ATTACH_COMPLETE }

```

CUDA Kernel Events.

- enum CUDBGException_t {

```

CUDBG_EXCEPTION_UNKNOWN,
CUDBG_EXCEPTION_NONE,
CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS,
CUDBG_EXCEPTION_LANE_USER_STACK_OVERFLOW,
CUDBG_EXCEPTION_DEVICE_HARDWARE_STACK_OVERFLOW,
CUDBG_EXCEPTION_WARP_ILLEGAL_INSTRUCTION,
CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS,
CUDBG_EXCEPTION_WARP_MISALIGNED_ADDRESS,
CUDBG_EXCEPTION_WARP_INVALID_ADDRESS_SPACE,
CUDBG_EXCEPTION_WARP_INVALID_PC,
CUDBG_EXCEPTION_WARP_HARDWARE_STACK_OVERFLOW,
CUDBG_EXCEPTION_DEVICE_ILLEGAL_ADDRESS,
CUDBG_EXCEPTION_LANE_MISALIGNED_ADDRESS }

```

Hardware Exception Types.

- enum CUDBGGridStatus {

```

CUDBG_GRID_STATUS_INVALID,
CUDBG_GRID_STATUS_PENDING,
CUDBG_GRID_STATUS_ACTIVE,
CUDBG_GRID_STATUS_SLEEPING,
CUDBG_GRID_STATUS_TERMINATED,
CUDBG_GRID_STATUS_UNDETERMINED }

```

Grid status.

- enum CUDBGKernelType {

```

CUDBG_KNL_TYPE_UNKNOWN,
CUDBG_KNL_TYPE_SYSTEM,
CUDBG_KNL_TYPE_APPLICATION }

```

Kernel types.

- enum CUDBGRegClass {

```

REG_CLASS_INVALID,
REG_CLASS_REG_CC,
REG_CLASS_REG_PRED,
REG_CLASS_REG_ADDR,
REG_CLASS_REG_HALF,

```

```
REG_CLASS_REG_FULL,  
REG_CLASS_MEM_LOCAL,  
REG_CLASS_LMEM_REG_OFFSET }
```

Physical register types.

- `enum CUDBGResult {`
 `CUDBG_SUCCESS,`
 `CUDBG_ERROR_UNKNOWN,`
 `CUDBG_ERROR_BUFFER_TOO_SMALL,`
 `CUDBG_ERROR_UNKNOWN_FUNCTION,`
 `CUDBG_ERROR_INVALID_ARGS,`
 `CUDBG_ERROR_UNINITIALIZED,`
 `CUDBG_ERROR_INVALID_COORDINATES,`
 `CUDBG_ERROR_INVALID_MEMORY_SEGMENT,`
 `CUDBG_ERROR_INVALID_MEMORY_ACCESS,`
 `CUDBG_ERROR_MEMORY_MAPPING_FAILED,`
 `CUDBG_ERROR_INTERNAL,`
 `CUDBG_ERROR_INVALID_DEVICE,`
 `CUDBG_ERROR_INVALID_SM,`
 `CUDBG_ERROR_INVALID_WARP,`
 `CUDBG_ERROR_INVALID_LANE,`
 `CUDBG_ERROR_SUSPENDED_DEVICE,`
 `CUDBG_ERROR_RUNNING_DEVICE,`
 `CUDBG_ERROR_INVALID_ADDRESS,`
 `CUDBG_ERROR_INCOMPATIBLE_API,`
 `CUDBG_ERROR_INITIALIZATION_FAILURE,`
 `CUDBG_ERROR_INVALID_GRID,`
 `CUDBG_ERROR_NO_EVENT_AVAILABLE,`
 `CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED,`
 `CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED,`
 `CUDBG_ERROR_INVALID_ATTRIBUTE,`
 `CUDBG_ERROR_ZERO_CALL_DEPTH,`
 `CUDBG_ERROR_INVALID_CALL_LEVEL,`
 `CUDBG_ERROR_COMMUNICATION_FAILURE,`
 `CUDBG_ERROR_INVALID_CONTEXT,`
 `CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM ,`
 `CUDBG_ERROR_OS_RESOURCES,`
 `CUDBG_ERROR_FORK_FAILED }`

Result values of all the API routines.

Functions

- [`CUDBGResult cudbgGetAPIVersion`](#) (uint32_t *major, uint32_t *minor, uint32_t *rev)

Get the API version supported by the CUDA driver.

7.1.1 Detailed Description

Header file for the CUDA debugger API.

7.1.2 Enumeration Type Documentation

7.1.2.1 enum CUDBGAttribute

Query attribute.

Enumerator:

`CUDBG_ATTR_GRID_LAUNCH_BLOCKING` whether the launch is synchronous or not.

`CUDBG_ATTR_GRID_TID` The id of the host thread that launched the grid.

7.1.2.2 enum CUDBGException_t

Hardware Exception Types.

Enumerator:

`CUDBG_EXCEPTION_UNKNOWN` Reported if we do not know what exception the chip has hit (global error).

`CUDBG_EXCEPTION_NONE` Reported when there is no exception on the chip (no error).

`CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS` Reported when memcheck(enabled within cuda-gdb) finds access violations (lane error: precise software generated exception).

`CUDBG_EXCEPTION_LANE_USER_STACK_OVERFLOW` Reported from user (data) stack overflow checks in each function's prologue (lane error: precise software generated exception, ABI-only).

`CUDBG_EXCEPTION_DEVICE_HARDWARE_STACK_OVERFLOW` Reported if CRS overflows (global error: the warp that caused this will terminate).

`CUDBG_EXCEPTION_WARP_ILLEGAL_INSTRUCTION` Reported when any lane in a warp executes an illegal instruction (warp error: invalid branch target, invalid opcode, misaligned/oor reg, invalid immediates, etc.).

`CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS` Reported when any lane in a warp accesses memory that is out of range (warp error: lmem_lo/hi, shared, and 40-bit va accesses).

`CUDBG_EXCEPTION_WARP_MISALIGNED_ADDRESS` Reported when any lane in a warp accesses memory that is misaligned (warp error: lmem_lo/hi, shared, and 40-bit va accesses).

`CUDBG_EXCEPTION_WARP_INVALID_ADDRESS_SPACE` Reported when any lane in a warp executes an instruction that accesses a memory space that is not permitted for that instruction (warp error).

`CUDBG_EXCEPTION_WARP_INVALID_PC` Reported when any lane in a warp advances its PC beyond the 32-bit address space (warp error).

`CUDBG_EXCEPTION_WARP_HARDWARE_STACK_OVERFLOW` Reported when any lane in a warp hits (uncommon) stack issues (warp error: stack error or api stack overflow).

CUDBG_EXCEPTION_DEVICE_ILLEGAL_ADDRESS Reported when MMU detects an error (global error: L1 error status field is set in the global esr – for the most part this catches errors SM couldn't catch with oor address detection).

CUDBG_EXCEPTION_LANE_MISALIGNED_ADDRESS Reported when memcheck(enabled within cuda-gdb) finds access violations (lane error: precise software generated exception).

7.1.2.3 enum CUDBGGridStatus

Grid status.

Enumerator:

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.

7.1.2.4 enum CUDBGKernelType

Kernel types.

Enumerator:

CUDBG_KNL_TYPE_UNKNOWN Unknown kernel type. Fall-back value.

CUDBG_KNL_TYPE_SYSTEM System kernel, launched by the CUDA driver (cudaMemset, ...).

CUDBG_KNL_TYPE_APPLICATION Application kernel, launched by the application.

7.1.2.5 enum CUDBGRegClass

Physical register types.

Enumerator:

REG_CLASS_INVALID The physical register is invalid.

REG_CLASS_REG_CC The physical register is a condition code register. Unused.

REG_CLASS_REG_PRED The physical register is a predicate register. Unused.

REG_CLASS_REG_ADDR The physical register is an address register. Unused.

REG_CLASS_REG_HALF The physical register is a 16-bit register. Unused.

REG_CLASS_REG_FULL The physical register is a 32-bit register.

REG_CLASS_MEM_LOCAL The content of the physical register has been spilled to memory.

REG_CLASS_LMEM_REG_OFFSET The content of the physical register has been spilled to the local stack (ABI only).

7.1.2.6 enum CUDBGResult

Result values of all the API routines.

Enumerator:

- CUDBG_SUCCESS* The API call executed successfully.
- CUDBG_ERROR_UNKNOWN* Error type not listed below.
- CUDBG_ERROR_BUFFER_TOO_SMALL* Cannot copy all the queried data into the buffer argument.
- CUDBG_ERROR_UNKNOWN_FUNCTION* Function cannot be found in the CUDA kernel.
- CUDBG_ERROR_INVALID_ARGS* Wrong use of arguments (NULL pointer, illegal value,...).
- CUDBG_ERROR_UNINITIALIZED* Debugger API has not yet been properly initialized.
- CUDBG_ERROR_INVALID_COORDINATES* Invalid block or thread coordinates were provided.
- CUDBG_ERROR_INVALID_MEMORY_SEGMENT* Invalid memory segment requested.
- CUDBG_ERROR_INVALID_MEMORY_ACCESS* Requested address (+size) is not within proper segment boundaries.
- CUDBG_ERROR_MEMORY_MAPPING_FAILED* Memory is not mapped and cannot be mapped.
- CUDBG_ERROR_INTERNAL* A debugger internal error occurred.
- CUDBG_ERROR_INVALID_DEVICE* Specified device cannot be found.
- CUDBG_ERROR_INVALID_SM* Specified sm cannot be found.
- CUDBG_ERROR_INVALID_WARP* Specified warp cannot be found.
- CUDBG_ERROR_INVALID_LANE* Specified lane cannot be found.
- CUDBG_ERROR_SUSPENDED_DEVICE* The requested operation is not allowed when the device is suspended.
- CUDBG_ERROR_RUNNING_DEVICE* Device is running and not suspended.
- CUDBG_ERROR_INVALID_ADDRESS* Address is out-of-range.
- CUDBG_ERROR_INCOMPATIBLE_API* The requested API is not available.
- CUDBG_ERROR_INITIALIZATION_FAILURE* The API could not be initialized.
- CUDBG_ERROR_INVALID_GRID* The specified grid is not valid.
- CUDBG_ERROR_NO_EVENT_AVAILABLE* The event queue is empty and there is no event left to be processed.
- CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED* Some devices were excluded because they have a watchdog associated with them.
- CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED* All devices were excluded because they have a watchdog associated with them.
- CUDBG_ERROR_INVALID_ATTRIBUTE* Specified attribute does not exist or is incorrect.
- CUDBG_ERROR_ZERO_CALL_DEPTH* No function calls have been made on the device.
- CUDBG_ERROR_INVALID_CALL_LEVEL* Specified call level is invalid.
- CUDBG_ERROR_COMMUNICATION_FAILURE* Communication error between the debugger and the application.
- CUDBG_ERROR_INVALID_CONTEXT* Specified context cannot be found.
- CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM* Requested address was not originally allocated from device memory (most likely visible in system memory).
- CUDBG_ERROR_OS_RESOURCES* Error while allocating resources from the OS.
- CUDBG_ERROR_FORK_FAILED* Error while forking the debugger process.

7.1.3 Function Documentation

7.1.3.1 CUDBGResult cudbgGetAPIVersion (uint32_t * *major*, uint32_t * *minor*, uint32_t * *rev*)

Get the API version supported by the CUDA driver.

Parameters:

major - the major version number

minor - the minor version number

rev - the revision version number

Returns:

CUDBG_ERROR_INVALID_ARGS,
CUDBG_SUCCESS

See also:

cudbgGetAPI

Index

- acknowledgeEvent30
 - EVENT, [58](#)
- acknowledgeEvents42
 - EVENT, [58](#)
- acknowledgeSyncEvents
 - EVENT, [58](#)
- BP
 - setBreakpoint, [16](#)
 - setBreakpoint31, [16](#)
 - unsetBreakpoint, [17](#)
 - unsetBreakpoint31, [17](#)
- Breakpoints, [16](#)
- clearAttachState
 - CUDBGAPI_st, [67](#)
- cudaDebugger.h, [103](#)
 - CUDBG_ATTR_GRID_LAUNCH_BLOCKING, [107](#)
 - CUDBG_ATTR_GRID_TID, [107](#)
 - CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM, [109](#)
 - CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED, [109](#)
 - CUDBG_ERROR_BUFFER_TOO_SMALL, [109](#)
 - CUDBG_ERROR_COMMUNICATION_FAILURE, [109](#)
 - CUDBG_ERROR_FORK_FAILED, [109](#)
 - CUDBG_ERROR_INCOMPATIBLE_API, [109](#)
 - CUDBG_ERROR_INITIALIZATION_FAILURE, [109](#)
 - CUDBG_ERROR_INTERNAL, [109](#)
 - CUDBG_ERROR_INVALID_ADDRESS, [109](#)
 - CUDBG_ERROR_INVALID_ARGS, [109](#)
 - CUDBG_ERROR_INVALID_ATTRIBUTE, [109](#)
 - CUDBG_ERROR_INVALID_CALL_LEVEL, [109](#)
 - CUDBG_ERROR_INVALID_CONTEXT, [109](#)
 - CUDBG_ERROR_INVALID_COORDINATES, [109](#)
 - CUDBG_ERROR_INVALID_DEVICE, [109](#)
 - CUDBG_ERROR_INVALID_GRID, [109](#)
 - CUDBG_ERROR_INVALID_LANE, [109](#)
 - CUDBG_ERROR_INVALID_MEMORY_ACCESS, [109](#)
 - CUDBG_ERROR_INVALID_MEMORY_SEGMENT, [109](#)
 - CUDBG_ERROR_INVALID_SM, [109](#)
 - CUDBG_ERROR_INVALID_WARP, [109](#)
 - CUDBG_ERROR_MEMORY_MAPPING_FAILED, [109](#)
 - CUDBG_ERROR_NO_EVENT_AVAILABLE, [109](#)
 - CUDBG_ERROR_OS_RESOURCES, [109](#)
 - CUDBG_ERROR_RUNNING_DEVICE, [109](#)
 - CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED, [109](#)
 - CUDBG_ERROR_SUSPENDED_DEVICE, [109](#)
 - CUDBG_ERROR_UNINITIALIZED, [109](#)
 - CUDBG_ERROR_UNKNOWN, [109](#)
 - CUDBG_ERROR_UNKNOWN_FUNCTION, [109](#)
 - CUDBG_ERROR_ZERO_CALL_DEPTH, [109](#)
 - CUDBG_EXCEPTION_DEVICE_HARDWARE_STACK_OVERFLOW, [107](#)
 - CUDBG_EXCEPTION_DEVICE_ILLEGAL_ADDRESS, [107](#)
 - CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS, [107](#)
 - CUDBG_EXCEPTION_LANE_MISALIGNED_ADDRESS, [108](#)
 - CUDBG_EXCEPTION_LANE_USER_STACK_OVERFLOW, [107](#)
 - CUDBG_EXCEPTION_NONE, [107](#)
 - CUDBG_EXCEPTION_UNKNOWN, [107](#)
 - CUDBG_EXCEPTION_WARP_HARDWARE_STACK_OVERFLOW, [107](#)
 - CUDBG_EXCEPTION_WARP_ILLEGAL_INSTRUCTION, [107](#)
 - CUDBG_EXCEPTION_WARP_INVALID_ADDRESS_SPACE, [107](#)
 - CUDBG_EXCEPTION_WARP_INVALID_PC, [107](#)
 - CUDBG_EXCEPTION_WARP_MISALIGNED_ADDRESS, [107](#)
 - CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS, [107](#)
 - CUDBG_GRID_STATUS_ACTIVE, [108](#)
 - CUDBG_GRID_STATUS_INVALID, [108](#)
 - CUDBG_GRID_STATUS_PENDING, [108](#)
 - CUDBG_GRID_STATUS_SLEEPING, [108](#)
 - CUDBG_GRID_STATUS_TERMINATED, [108](#)
 - CUDBG_GRID_STATUS_UNDETERMINED, [108](#)
 - CUDBG_KNL_TYPE_APPLICATION, [108](#)

- CUDBG_KNL_TYPE_SYSTEM, [108](#)
- CUDBG_KNL_TYPE_UNKNOWN, [108](#)
- CUDBG_SUCCESS, [109](#)
- CUDBGAttribute, [107](#)
- CUDBGException_t, [107](#)
- cudbgGetAPIVersion, [110](#)
- CUDBGGridStatus, [108](#)
- CUDBGKernelType, [108](#)
- CUDBGRegClass, [108](#)
- CUDBGResult, [108](#)
- REG_CLASS_INVALID, [108](#)
- REG_CLASS_LMEM_REG_OFFSET, [108](#)
- REG_CLASS_MEM_LOCAL, [108](#)
- REG_CLASS_REG_ADDR, [108](#)
- REG_CLASS_REG_CC, [108](#)
- REG_CLASS_REG_FULL, [108](#)
- REG_CLASS_REG_HALF, [108](#)
- REG_CLASS_REG_PRED, [108](#)
- CUDBG_ATTR_GRID_LAUNCH_BLOCKING
cudadebugger.h, [107](#)
- CUDBG_ATTR_GRID_TID
cudadebugger.h, [107](#)
- CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_-
MEM
cudadebugger.h, [109](#)
- CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED
cudadebugger.h, [109](#)
- CUDBG_ERROR_BUFFER_TOO_SMALL
cudadebugger.h, [109](#)
- CUDBG_ERROR_COMMUNICATION_FAILURE
cudadebugger.h, [109](#)
- CUDBG_ERROR_FORK_FAILED
cudadebugger.h, [109](#)
- CUDBG_ERROR_INCOMPATIBLE_API
cudadebugger.h, [109](#)
- CUDBG_ERROR_INITIALIZATION_FAILURE
cudadebugger.h, [109](#)
- CUDBG_ERROR_INTERNAL
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_ADDRESS
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_ARGS
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_ATTRIBUTE
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_CALL_LEVEL
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_CONTEXT
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_COORDINATES
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_DEVICE
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_GRID
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_LANE
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_MEMORY_ACCESS
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_MEMORY_SEGMENT
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_SM
cudadebugger.h, [109](#)
- CUDBG_ERROR_INVALID_WARP
cudadebugger.h, [109](#)
- CUDBG_ERROR_MEMORY_MAPPING_FAILED
cudadebugger.h, [109](#)
- CUDBG_ERROR_NO_EVENT_AVAILABLE
cudadebugger.h, [109](#)
- CUDBG_ERROR_OS_RESOURCES
cudadebugger.h, [109](#)
- CUDBG_ERROR_RUNNING_DEVICE
cudadebugger.h, [109](#)
- CUDBG_ERROR_SOME_DEVICES_-
WATCHDOGGED
cudadebugger.h, [109](#)
- CUDBG_ERROR_SUSPENDED_DEVICE
cudadebugger.h, [109](#)
- CUDBG_ERROR_UNINITIALIZED
cudadebugger.h, [109](#)
- CUDBG_ERROR_UNKNOWN
cudadebugger.h, [109](#)
- CUDBG_ERROR_UNKNOWN_FUNCTION
cudadebugger.h, [109](#)
- CUDBG_ERROR_ZERO_CALL_DEPTH
cudadebugger.h, [109](#)
- CUDBG_EVENT_ATTACH_COMPLETE
EVENT, [58](#)
- CUDBG_EVENT_CTX_CREATE
EVENT, [57](#)
- CUDBG_EVENT_CTX_DESTROY
EVENT, [57](#)
- CUDBG_EVENT_CTX_POP
EVENT, [57](#)
- CUDBG_EVENT_CTX_PUSH
EVENT, [57](#)
- CUDBG_EVENT_ELF_IMAGE_LOADED
EVENT, [57](#)
- CUDBG_EVENT_INTERNAL_ERROR
EVENT, [57](#)
- CUDBG_EVENT_INVALID
EVENT, [57](#)
- CUDBG_EVENT_KERNEL_FINISHED
EVENT, [57](#)
- CUDBG_EVENT_KERNEL_READY
EVENT, [57](#)
- CUDBG_EVENT_TIMEOUT
EVENT, [57](#)

- CUDBG_EXCEPTION_DEVICE_HARDWARE_-
STACK_OVERFLOW
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_DEVICE_ILLEGAL_-
ADDRESS
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_LANE_ILLEGAL_ADDRESS
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_LANE_MISALIGNED_-
ADDRESS
 cudadebugger.h, [108](#)
- CUDBG_EXCEPTION_LANE_USER_STACK_-
OVERFLOW
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_NONE
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_UNKNOWN
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_WARP_HARDWARE_-
STACK_OVERFLOW
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_WARP_ILLEGAL_-
INSTRUCTION
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_WARP_INVALID_-
ADDRESS_SPACE
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_WARP_INVALID_PC
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_WARP_MISALIGNED_-
ADDRESS
 cudadebugger.h, [107](#)
- CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_-
ADDRESS
 cudadebugger.h, [107](#)
- CUDBG_GRID_STATUS_ACTIVE
 cudadebugger.h, [108](#)
- CUDBG_GRID_STATUS_INVALID
 cudadebugger.h, [108](#)
- CUDBG_GRID_STATUS_PENDING
 cudadebugger.h, [108](#)
- CUDBG_GRID_STATUS_SLEEPING
 cudadebugger.h, [108](#)
- CUDBG_GRID_STATUS_TERMINATED
 cudadebugger.h, [108](#)
- CUDBG_GRID_STATUS_UNDETERMINED
 cudadebugger.h, [108](#)
- CUDBG_KNL_TYPE_APPLICATION
 cudadebugger.h, [108](#)
- CUDBG_KNL_TYPE_SYSTEM
 cudadebugger.h, [108](#)
- CUDBG_KNL_TYPE_UNKNOWN
 cudadebugger.h, [108](#)
- CUDBG_SUCCESS
 cudadebugger.h, [109](#)
- CUDBGAPI_st, [61](#)
 - clearAttachState, [67](#)
 - getGridStatus, [67](#)
 - requestCleanupOnDetach, [67](#)
- CUDBGAttribute
 cudadebugger.h, [107](#)
- CUDBGEvent, [68](#)
- CUDBGEvent30, [69](#)
 - CUDBGEvent30::cases30_st, [70](#)
 - CUDBGEvent30::cases30_st::elfImageLoaded30_st, [71](#)
 - CUDBGEvent30::cases30_st::kernelFinished30_st, [72](#)
 - CUDBGEvent30::cases30_st::kernelReady30_st, [73](#)
- CUDBGEvent32, [74](#)
 - CUDBGEvent32::cases32_st, [75](#)
 - CUDBGEvent32::cases32_st::contextCreate32_st, [76](#)
 - CUDBGEvent32::cases32_st::contextDestroy32_st, [77](#)
 - CUDBGEvent32::cases32_st::contextPop32_st, [78](#)
 - CUDBGEvent32::cases32_st::contextPush32_st, [79](#)
 - CUDBGEvent32::cases32_st::elfImageLoaded32_st, [80](#)
 - CUDBGEvent32::cases32_st::kernelFinished32_st, [81](#)
 - CUDBGEvent32::cases32_st::kernelReady32_st, [82](#)
- CUDBGEvent42, [83](#)
 - CUDBGEvent42::cases42_st, [84](#)
 - CUDBGEvent42::cases42_st::contextCreate42_st, [85](#)
 - CUDBGEvent42::cases42_st::contextDestroy42_st, [86](#)
 - CUDBGEvent42::cases42_st::contextPop42_st, [87](#)
 - CUDBGEvent42::cases42_st::contextPush42_st, [88](#)
 - CUDBGEvent42::cases42_st::elfImageLoaded42_st, [89](#)
 - CUDBGEvent42::cases42_st::kernelFinished42_st, [90](#)
 - CUDBGEvent42::cases42_st::kernelReady42_st, [91](#)
- CUDBGEvent::cases_st, [92](#)
 - CUDBGEvent::cases_st::contextCreate_st, [93](#)
 - CUDBGEvent::cases_st::contextDestroy_st, [94](#)
 - CUDBGEvent::cases_st::contextPop_st, [95](#)
 - CUDBGEvent::cases_st::contextPush_st, [96](#)
 - CUDBGEvent::cases_st::elfImageLoaded_st, [97](#)
 - CUDBGEvent::cases_st::internalError_st, [98](#)
 - CUDBGEvent::cases_st::kernelFinished_st, [99](#)
 - CUDBGEvent::cases_st::kernelReady_st, [100](#)
- CUDBGEventCallbackData, [101](#)
- CUDBGEventCallbackData40, [102](#)
- CUDBGEventKind
 EVENT, [57](#)
- CUDBGException_t
 cudadebugger.h, [107](#)
- cudbgGetAPIVersion
 cudadebugger.h, [110](#)
- CUDBGGridStatus
 cudadebugger.h, [108](#)
- CUDBGKernelType
 cudadebugger.h, [108](#)
- CUDBGRegClass
 cudadebugger.h, [108](#)

- CUDBGResult
 - [cudadebugger.h](#), 108
- DEV
 - [getDeviceType](#), 48
 - [getNumDevices](#), 48
 - [getNumLanes](#), 49
 - [getNumRegisters](#), 49
 - [getNumSMs](#), 50
 - [getNumWarps](#), 50
 - [getSmType](#), 51
- Device Execution Control, 13
- Device Properties, 48
- Device State Alteration, 39
- Device State Inspection, 18
- disassemble
 - [DWARF](#), 52
- DWARF
 - [disassemble](#), 52
 - [getHostAddrFromDeviceAddr](#), 52
 - [getPhysicalRegister30](#), 53
 - [getPhysicalRegister40](#), 53
 - [isDeviceCodeAddress](#), 54
 - [lookupDeviceCodeSymbol](#), 54
- DWARF Utilities, 52
- EVENT
 - [acknowledgeEvent30](#), 58
 - [acknowledgeEvents42](#), 58
 - [acknowledgeSyncEvents](#), 58
 - [CUDBG_EVENT_ATTACH_COMPLETE](#), 58
 - [CUDBG_EVENT_CTX_CREATE](#), 57
 - [CUDBG_EVENT_CTX_DESTROY](#), 57
 - [CUDBG_EVENT_CTX_POP](#), 57
 - [CUDBG_EVENT_CTX_PUSH](#), 57
 - [CUDBG_EVENT_ELF_IMAGE_LOADED](#), 57
 - [CUDBG_EVENT_INTERNAL_ERROR](#), 57
 - [CUDBG_EVENT_INVALID](#), 57
 - [CUDBG_EVENT_KERNEL_FINISHED](#), 57
 - [CUDBG_EVENT_KERNEL_READY](#), 57
 - [CUDBG_EVENT_TIMEOUT](#), 57
 - [CUDBGEventKind](#), 57
 - [getNextAsyncEvent](#), 58
 - [getNextEvent30](#), 58
 - [getNextEvent32](#), 59
 - [getNextEvent42](#), 59
 - [getNextSyncEvent](#), 59
 - [setNotifyNewEventCallback](#), 59
 - [setNotifyNewEventCallback31](#), 60
 - [setNotifyNewEventCallback40](#), 60
- Events, 55
- EXEC
 - [resumeDevice](#), 13
 - [singleStepWarp](#), 13
- [singleStepWarp40](#), 14
- [suspendDevice](#), 14
- finalize
 - [INIT](#), 11
- [getBlockDim](#)
 - [GRID](#), 44
- [getDeviceType](#)
 - [DEV](#), 48
- [getElfImage](#)
 - [GRID](#), 45
- [getElfImage32](#)
 - [GRID](#), 45
- [getGridAttribute](#)
 - [GRID](#), 45
- [getGridAttributes](#)
 - [GRID](#), 46
- [getGridDim](#)
 - [GRID](#), 46
- [getGridDim32](#)
 - [GRID](#), 47
- [getGridStatus](#)
 - [CUDBGAPI_st](#), 67
- [getHostAddrFromDeviceAddr](#)
 - [DWARF](#), 52
- [getNextAsyncEvent](#)
 - [EVENT](#), 58
- [getNextEvent30](#)
 - [EVENT](#), 58
- [getNextEvent32](#)
 - [EVENT](#), 59
- [getNextEvent42](#)
 - [EVENT](#), 59
- [getNextSyncEvent](#)
 - [EVENT](#), 59
- [getNumDevices](#)
 - [DEV](#), 48
- [getNumLanes](#)
 - [DEV](#), 49
- [getNumRegisters](#)
 - [DEV](#), 49
- [getNumSMs](#)
 - [DEV](#), 50
- [getNumWarps](#)
 - [DEV](#), 50
- [getPhysicalRegister30](#)
 - [DWARF](#), 53
- [getPhysicalRegister40](#)
 - [DWARF](#), 53
- [getSmType](#)
 - [DEV](#), 51
- [getTID](#)
 - [GRID](#), 47

GRID

- getBlockDim, [44](#)
- getElfImage, [45](#)
- getElfImage32, [45](#)
- getGridAttribute, [45](#)
- getGridAttributes, [46](#)
- getGridDim, [46](#)
- getGridDim32, [47](#)
- getTID, [47](#)

- Grid Properties, [44](#)

INIT

- finalize, [11](#)
- initialize, [11](#)

- Initialization, [11](#)

- initialize

- INIT, [11](#)

- isDeviceCodeAddress

- DWARF, [54](#)

- lookupDeviceCodeSymbol

- DWARF, [54](#)

- memcheckReadErrorAddress

- READ, [20](#)

READ

- memcheckReadErrorAddress, [20](#)
- readActiveLanes, [20](#)
- readBlockIdx, [21](#)
- readBlockIdx32, [22](#)
- readBrokenWarps, [22](#)
- readCallDepth, [23](#)
- readCallDepth32, [23](#)
- readCodeMemory, [24](#)
- readConstMemory, [24](#)
- readGlobalMemory, [25](#)
- readGlobalMemory31, [25](#)
- readGridId, [26](#)
- readLaneException, [27](#)
- readLaneStatus, [27](#)
- readLocalMemory, [27](#)
- readParamMemory, [28](#)
- readPC, [29](#)
- readPinnedMemory, [29](#)
- readRegister, [30](#)
- readReturnAddress, [31](#)
- readReturnAddress32, [31](#)
- readSharedMemory, [32](#)
- readSyscallCallDepth, [32](#)
- readTextureMemory, [33](#)
- readTextureMemoryBindless, [34](#)
- readThreadIdx, [34](#)
- readValidLanes, [35](#)
- readValidWarps, [36](#)

- readVirtualPC, [36](#)

- readVirtualReturnAddress, [37](#)

- readVirtualReturnAddress32, [37](#)

- writePinnedMemory, [38](#)

- readActiveLanes

- READ, [20](#)

- readBlockIdx

- READ, [21](#)

- readBlockIdx32

- READ, [22](#)

- readBrokenWarps

- READ, [22](#)

- readCallDepth

- READ, [23](#)

- readCallDepth32

- READ, [23](#)

- readCodeMemory

- READ, [24](#)

- readConstMemory

- READ, [24](#)

- readGlobalMemory

- READ, [25](#)

- readGlobalMemory31

- READ, [25](#)

- readGridId

- READ, [26](#)

- readLaneException

- READ, [27](#)

- readLaneStatus

- READ, [27](#)

- readLocalMemory

- READ, [27](#)

- readParamMemory

- READ, [28](#)

- readPC

- READ, [29](#)

- readPinnedMemory

- READ, [29](#)

- readRegister

- READ, [30](#)

- readReturnAddress

- READ, [31](#)

- readReturnAddress32

- READ, [31](#)

- readSharedMemory

- READ, [32](#)

- readSyscallCallDepth

- READ, [32](#)

- readTextureMemory

- READ, [33](#)

- readTextureMemoryBindless

- READ, [34](#)

- readThreadIdx

- READ, [34](#)

- readValidLanes
 - READ, [35](#)
- readValidWarps
 - READ, [36](#)
- readVirtualPC
 - READ, [36](#)
- readVirtualReturnAddress
 - READ, [37](#)
- readVirtualReturnAddress32
 - READ, [37](#)
- REG_CLASS_INVALID
 - cudadebugger.h, [108](#)
- REG_CLASS_LMEM_REG_OFFSET
 - cudadebugger.h, [108](#)
- REG_CLASS_MEM_LOCAL
 - cudadebugger.h, [108](#)
- REG_CLASS_REG_ADDR
 - cudadebugger.h, [108](#)
- REG_CLASS_REG_CC
 - cudadebugger.h, [108](#)
- REG_CLASS_REG_FULL
 - cudadebugger.h, [108](#)
- REG_CLASS_REG_HALF
 - cudadebugger.h, [108](#)
- REG_CLASS_REG_PRED
 - cudadebugger.h, [108](#)
- requestCleanupOnDetach
 - CUDBGAPI_st, [67](#)
- resumeDevice
 - EXEC, [13](#)
- setBreakpoint
 - BP, [16](#)
- setBreakpoint31
 - BP, [16](#)
- setNotifyNewEventCallback
 - EVENT, [59](#)
- setNotifyNewEventCallback31
 - EVENT, [60](#)
- setNotifyNewEventCallback40
 - EVENT, [60](#)
- singleStepWarp
 - EXEC, [13](#)
- singleStepWarp40
 - EXEC, [14](#)
- suspendDevice
 - EXEC, [14](#)
- unsetBreakpoint
 - BP, [17](#)
- unsetBreakpoint31
 - BP, [17](#)
- WRITE
 - writeGlobalMemory, [39](#)
- writeGlobalMemory31, [40](#)
- writeLocalMemory, [40](#)
- writeParamMemory, [41](#)
- writeRegister, [41](#)
- writeSharedMemory, [42](#)
- writeGlobalMemory
 - WRITE, [39](#)
- writeGlobalMemory31
 - WRITE, [40](#)
- writeLocalMemory
 - WRITE, [40](#)
- writeParamMemory
 - WRITE, [41](#)
- writePinnedMemory
 - READ, [38](#)
- writeRegister
 - WRITE, [41](#)
- writeSharedMemory
 - WRITE, [42](#)